IOx Application Hosting

This section contains the following topics:

Application Hosting

A hosted application is a software as a service solution, and it can be run remotely using commands. Application hosting gives administrators a platform for leveraging their own tools and utilities.

This chapter describes the Application Hosting feature and how to enable it.

Information About Application Hosting

This section provides information about Application Hosting.

Need for Application Hosting

The move to virtual environments has given rise to the need to build applications that are reusable, portable, and scalable. Application hosting gives administrators a platform for leveraging their own tools and utilities. An application, hosted on a network device, can serve a variety of purposes. This ranges from automation, configuration management monitoring, and integration with existing tool chains.

Cisco devices support third-party off-the-shelf applications built using Linux tool chains. Users can run custom applications cross-compiled with the software development kit that Cisco provides.

IOx Overview

IOx is a Cisco-developed end-to-end application framework that provides application hosting capabilities for different application types on Cisco network platforms.

From Cisco IOS-XE Release 17.8.1, IOx installation on IR8340 requires Cisco supported mSATA to be the storage device. There are two partitions on the mSATA. One for IOS, and the other for IOx. The IOS partition will be mounted on **/mnt/msata** and IOx partition will be mounted on **/vol/harddisk**. OIR is not supported for mSATA device. When mSATA is inserted to the router, the router needs to be reloaded to have two partitions. If mSATA is not present, bootflash is used for application hosting.

In Cisco IOS XE Release 17.7.x, SD card is used as storage device for IOx. Any upgrade from 17.7.x to 17.8.x requires all applications to be reinstalled.

Cisco Application Hosting Overview

The IR8340 allows you to deploy applications using the application hosting CLI commands. You can also deploy applications using the Local Manager.

Application hosting provides the following services:

- Launches designated applications in containers.
- Checks available resources (memory, CPU, and storage), and allocates and manages them.
- Provides support for console logging.
- Provides a CLI endpoint.
- Provides an application hosting infrastructure referred to as Cisco Application Framework (CAF).
- Helps in the setup of platform-specific networking (packet-path) via VirtualPortGroup and management interfaces.

The container is referred to as the virtualization environment provided to run the guest application on the host operating system. The Cisco IOS-XE virtualization services provide manageability and networking models for running guest applications. The virtualization infrastructure allows the administrator to define a logical interface that specifies the connectivity between the host and the guest. IOx maps the logical interface into the Virtual Network Interface Card (vNIC) that the guest application uses.

Applications to be deployed in the containers are packaged as TAR files. The configuration that is specific to these applications is also packaged as part of the TAR file.

The management interface on the device connects the application hosting network to the IOS management interface. The Layer 3 interface of the application receives the Layer 2 bridged traffic from the IOS management interface. The management interface connects through the management bridge to the container/application interface. The IP address of the application must be on the same subnet as the management interface IP address.

IOXMAN

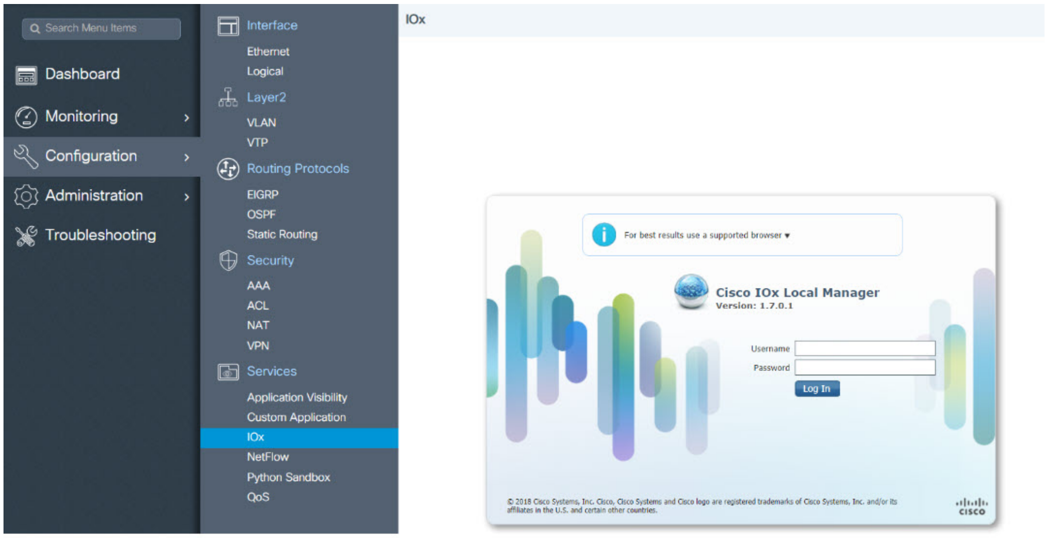
IOXMAN is a process that establishes a tracing infrastructure to provide logging or tracing services for guest applications, except Libvirt, that emulates serial devices. IOXMAN is based on the lifecycle of the guest application to enable and disable the tracing service, to send logging data to IOS syslog, to save tracing data to IOx tracelog, and to maintain IOx tracelog for each guest application.

Application Hosting on the IR8340 Router

This section describes the application hosting characteristics specific to the IR8340 router.

Application hosting can be achieved using the application hosting CLI commands as well as using Local Manager. Application hosting using Local Manager is done through WebUI. To deploy the applications using Local Manager, enable WebUI and then log in to Local Manager.

Figure 1: Local Manager



For best results use a supported browser ▾

© 2018 Cisco Systems, Inc. Cisco, Cisco Systems and Cisco logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

1. From WebUI, click on **Configuration** > **Services** > **IOx**

2. Log in using the username and password configured.

3. Follow the steps for the application lifecycle in the Cisco IOx Local Manager Reference Guide.

The next section explains the deployment of an application using the application hosting CLI commands.

Application Hosting on Layer 2 and Layer 3 Interfaces

The application configurations have two interfaces to support L2 and L3 traffic from the LAN and WAN ports respectively.

For application hosting, you can configure the L2 and L3 interfaces as following:

- L2 interfaces are configured with AppGigabitEthernet and VLAN with IP address in the same VLAN network, which are used or forwarding the L2 app traffic. Dedicated VLAN range 2340 - 2349 must be used for configuring L2 interfaces of application and to communicate the application for L2 traffic.

  You should configure the AppGigEthernet interface as a trunk interface.

- L3 interfaces or gateway interfaces are configured with Virtual port group, and IP address in the same network as VPG, which are used for forwarding the L3 traffic to applications.

VirtualPortGroup

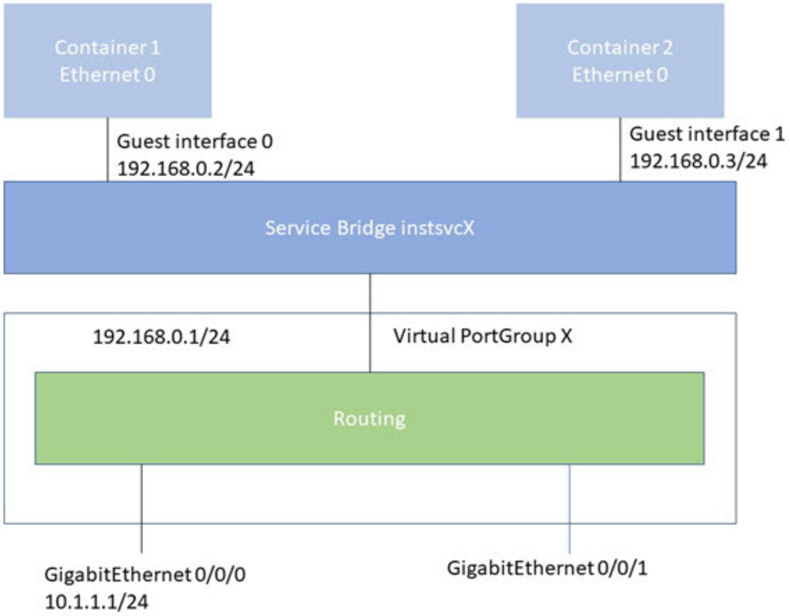
The VirtualPortGroup is a software construct on Cisco IOS that maps to a Linux bridge IP address. As such, the VirtualPortGroup represents the switch virtual interface (SVI) of the Linux container. Each bridge can contain multiple interfaces; each mapping to a different container. Each container can also have multiple interfaces.

VirtualPortGroup interfaces are configured by using the interface virtualportgroup command. Once these interfaces are created, IP address and other resources are allocated.

The VirtualPortGroup interface connects the application hosting network to the IOS routing domain. The Layer 3 interface of the application receives routed traffic from IOS. The VirtualPortGroup interface connects through the SVC Bridge to the container/application interface.

The following graphic helps to understand the relationship between the VirtualPortGroup and other interfaces.

Figure 2: Virtual Port Group Mapping



vNIC

For the container life cycle management, the Layer 3 routing model that supports one container per internal logical interface is used. This means that a virtual Ethernet pair is created for each application; and one interface of this pair, called vNIC is part of the application container. The other interface, called vpgX is part of the host system.

NIC is the standard Ethernet interface inside the container that connects to the platform dataplane for the sending and receiving of packets. IOx is responsible for the gateway (VirtualPortGroup interface), IP address, and unique MAC address assignment for each vNIC in the container.

The vNIC inside the container/application are considered as standard Ethernet interfaces.

How to Configure Application Hosting

The following sections provide information about the various tasks that comprise the configuration of application hosting.

Enabling IOx

Perform this task to enable access to the IOx Local Manager. The IOx Local Manager provides a web-based user interface that you can use to manage, administer, monitor, and troubleshoot apps on the host system, and to perform a variety of related activities.

Note In the steps that follow, IP HTTP commands do not enable IOx, but allow the user to access the WebUI to connect the IOx Local Manager.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**iox** **Example:** `Router(config)#iox`	Enables IOx.
Step 4	**ip http server** **Example:** `Router(config)#ip http server`	Enables the HTTP server on your IP or IPv6 system.
Step 5	**ip http secure-server** **Example:** `Router(config)#ip http secure-server`	Enables a secure HTTP (HTTPS) server.
Step 6	**username** *name* **privilege** *level* **secret** {**0** | **7** | *user-password*} *encrypted-password* **Example:** `Router(config)#username cisco privilege 15 secret 0 cisco`	Establishes a username-based authentication system and privilege level for the user. The username privilege level must be configured as 15.

	Command or Action	Purpose

Step 7	**end**
	Example:
	`Router(config)#end`
		Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Application Hosting to Layer 2 Interfaces

Follow these steps to configure application hosting to Layer 2 interfaces.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface AppGigEthernet** *number* **Example:** `Device(config)#interface AppGigabitEthernet 0/1/1`	Configures the AppGigabitEthernet and enters interface configuration mode.
Step 4	**switchport mode trunk** **Example:** `Device(config-if)#switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 5	**exit** **Example:** `Device(config-if)#exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iperf_3`	Configures the application and enters the application hosting configuration mode.
Step 7	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port for an application, and enters application-hosting trunk-configuration mode.

	Command or Action	Purpose
Step 8	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-app-hosting-trunk)# vlan 2340 guest-interface 1`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode.
Step 9	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:** `Device(config-app-hosting-vlan-access-ip)# guest-ipaddress 20.1.1.2 netmask 255.255.255.0`	Configures a static IP address.

Step 10	**end**
	Example:
	`Device(config-config-app-hosting-vlan-access-ip)# end`
		Exits application-hosting VLAN-access IP configuration mode and returns to privileged EXEC mode.

Configuring a VirtualPortGroup to a Layer 3 Data Port

Multiple Layer 3 data ports can be routed to one or more VirtualPortGroups or containers. VirutalPortGroups and Layer 3 data ports must be on different subnets.

Enable the **ip routing** command to allow external routing on the Layer 3 data-port.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)#ip routing`	Enables IP routing. The **ip routing** command must be enabled to allow external routing on Layer 3 data ports.
Step 4	**interface type number** **Example:** `Device(config)#interface gigabitethernet 0/0/0`	Configures an interface and enters interface configuration mode

	Command or Action	Purpose
Step 5	**no switchport** **Example:** `Device(config-if)#`**`no switchport`**	Places the interface in Layer 3 mode, and makes it operate more like a router interface rather than a switch port.
Step 6	**ip address ip-address mask** **Example:** `Device(config-if)#`**`ip address 10.1.1.1 255.255.255.0`**	Configures an IP address for the interface.
Step 7	**exit** **Example:** `Device(config-if)#`**`exit`**	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface type number** **Example:** `Device(config)#`**`interface virtualportgroup 0`**	Configures an interface and enters interface configuration mode.
Step 9	**ip address** *ip-address mask* **Example:** `Device(config-if)#`**`ip address 20.1.2.1 255.255.255.0`**	Configures an IP address for the interface.

Step 10	**end**
	Example:
	`Device(config-if)#`**`end`**
		Exits interface configuration mode and returns to privileged EXEC mode.

Step 11	**configure terminal** **Example:** `Device#`**`configure terminal`**	Enters global configuration mode.
Step 12	**interface vlan** *vlan-id* **Example:** `Device(config-if)#`**`interface vlan 2340`**	Configure the SVI interface for supporting L2 traffic. VLAN range: 2340 - 2349.
Step 13	**ip address** *ip-address mask* **Example:** `Device(config-if)#`**`ip address 20.1.1.1 255.255.255.0`**	Configures an IP address and IP subnet mask.

Step 14	**end**
	Example:
	`Device(config-if)#`**`end`**
		Exits interface configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 15	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 16	**app-hosting appid** *name* **Example:** `Device(config)#app-hosting appid iperf_3`	Configures the application and enters the application hosting configuration mode.
Step 17	**app-vnic gateway2 virtualportgroup 0 guest-interface 2** **Example:** `Device(config-app-hosting)#app-vnic gateway2 virtualportgroup 0 guest-interface 2`	Configures the application interface and the gateway of the application. You can create multiple interfaces with different virtualportgroups.
Step 18	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:** `Device(config-app-hosting-gateway0)#guest-ipaddress 20.1.2.2 netmask 255.255.255.0`	Configures the application Ethernet interface ip address.
Step 19	**app-default-gateway** *ip-address* **guest-interface 2** **Example:** `Device(config-app-hosting-gateway0)#app-default-gateway 20.1.2.1 guest-interface 2`	Configures the default gateway for the application. Only one gateway is supported.

Step 20	**end**
	Example:
	`Device#end`
		Exits global configuration mode and returns to privileged EXEC configuration mode.

Configuring Docker Run Time Options

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device >enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**app-hosting appid** *name* **Example:** `Router(config)#`**`app-hosting appid app1`**	Enables application hosting and enters application hosting configuration mode.
Step 4	**app-hosting docker** **Example:** `Device(config-app-hosting)# `**`app-resource docker`**	Enters application-hosting docker-configuration mode to specify application resource updates. Application start-up scripts are activated.

Step 5	**run-opts** *options*
	Example:
	`Device(config-app-hosting-docker)#`
	`run-opts 1 "-v $(APP_DATA):/data"`
		Specifies the Docker run time options.

Step 6	**end**
	Example:
	`Device(config-app-hosting-docker)# `**`end`**
		Exits application-hosting docker-configuration mode and returns to privileged EXEC mode.

Example

```
app-hosting appid app1
app-resource docker
run-opts 1 "--tmpfs /tmp:rw,size=128m"
```

Installing and Uninstalling Apps

Follow these steps to install or uninstall apps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device `**`>enable`**	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**app-hosting install appid** *application-name* **package** *package-path* **Example:** `Device# `**`app-hosting install appid lxc_app package flash:my_iox_app.tar`**	Installs an app from the specified location. The app can be installed from any local storage location such as, flash, bootflash, and usbflash0.
Step 3	**app-hosting activate appid** *application-name* **Example:**	Activates the application.

	Command or Action	Purpose
	`Device# app-hosting activate appid app1`	This command validates all application resource requests, and if all resources are available the application is activated; if not, the activation fails.
Step 4	**app-hosting start appid** *application-name* **Example:** `Device# app-hosting start appid app1`	Starts the application. Application start-up scripts are activated.
Step 5	**app-hosting stop appid** *application-name* **Example:** `Device#app-hosting stop appid app1`	Stops the application.
Step 6	**app-hosting deactivate appid** *application-name* **Example:** `Device# app-hosting deactivate appid app1`	Deactivates all resources allocated for the application.
Step 7	**app-hosting uninstall appid** *application-name* **Example:** `Device# app-hosting uninstall appid app1`	Uninstalls the application. Uninstalls all packaging and images stored. All changes and updates to the application are also removed.

What to do next

Note The app traffic to VirtualPortGroup interfaces will be blocked after you uninstall the app and reinstall it again with the same IP addresses, because the ARP entry for VirtualPortGroup interface is not updated after the app is reinstalled. You must clear the ARP cache for those IP addresses to be manually refreshed for the ARP.

Overriding the App Resource Configuration

Resource changes will take effect only after the app-hosting activate command is configured.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router>enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Router(config)#app-hosting appid app1`	Enables application hosting and enters application hosting configuration mode.
Step 4	**app-resource profile** *name* **Example:** `Router(config-app-hosting)#app-resource profile custom`	Configures the custom application resource profile, and enters custom application resource profile configuration mode. Only the custom profile name is supported.
Step 5	**cpu** *unit* **Example:** `Router(config-app-resource-profile-custom)# cpu 800`	Changes the default CPU allocation for the application. Resource values are application-specific, and any adjustment to these values must ensure that the application can run reliably with the changes.
Step 6	**memory** *memory* **Example:** `Router(config-app-resource-profile-custom)# memory 512`	Changes the default memory allocation.
Step 7	**vcpu** *number* **Example:** `Router(config-app-resource-profile-custom)# vcpu 2`	Changes the virtual CPU (vCPU) allocation for the application.

Step 8	**end**
	Example:
	`Router(config-app-resource-profile-custom)# end`
		Exits custom application resource profile configuration mode and returns to privileged EXEC mode.

Verifying the Application Hosting Configuration

1. **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Device>enable
```

2. show iox-service

Displays the status of all IOx services

Example:

```
Device# show iox-service
IOx Infrastructure Summary:
---------------------------
IOx service (CAF) : Running
IOx service (HA) : Not Supported
IOx service (IOxman) : Running
IOx service (Sec storage) : Running
Libvirtd 5.5.0 : Running
Dockerd 18.03.0 : Running
Device#
```

3. show app-hosting detail

Displays detailed information about the application.

Example:

```
Device#show app-hosting detail appid iperf_3
App id : iperf_3
Owner : iox
State : RUNNING
Application
Type : docker
Name : networkstatic/iperf3
Version : latest
Description :
Author : Brent
Path : bootflash:iperf3x86.tar
URL Path :
Activated profile name : custom

Resource reservation
Memory : 500 MB
Disk : 500 MB
CPU : 173 units
CPU-percent : 5 %
VCPU : 1

Platform resource profiles
Profile Name CPU(unit) Memory(MB) Disk(MB)
--------------------------------------------------------------

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:67:81:6f
IPv6 address : ::
Network name : mgmt-bridge300
eth3:
MAC address : 52:54:dd:b2:4d:86
IPv4 address : 20.1.2.2
IPv6 address : ::
Network name : VPG0
eth1:
MAC address : 52:54:dd:f2:29:67
IPv4 address : 20.1.1.2
```

```
IPv6 address : 2001:1::5054:ddff:fef2:2967
Network name : mgmt-bridge-v2340


Docker
------
Run-time information
Command :
Entry-point : /bin/sleep 10000
Run options in use : --entrypoint '/bin/sleep 10000'
Package run options :
Application health information
Status : 0
Last probe error :
Last probe output :

Device#
```

4. show app-hosting list

Displays the list of applications and their status.

Example:

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

IOx Configuration with ERSPAN

The traffic can be spanned to IOX applications with the ERSPAN configurations on LAN or WAN ports. ACL can be applied on traffic like ERSPAN with FSPAN.

Procedure

Step 1 Create ACL like any extended access-list.

Example:

```
ip access-list extended ACL120
10 permit ip host 120.1.1.1 host 120.120.120.120
```

Step 2 Configure ERSPAN session for LAN or WAN ports to span data to the application.

- Configure ERSPAN session for LAN ports to span data to the application.

  Note ERSPAN Session ID 1 - 4 are only supported on LAN ports.

  ```
  monitor session 1 type erspan-source
  source interface Gi0/1/10 rx
  filter access-group ACL120
  destination
  erspan-id 1
  ip address 20.1.2.2 <== Ip address of L2/VLAN interface on APP
  origin ip address 68.68.68.68
  ```

- configuring ERSPAN session for WAN ports to span data to the application.

```
monitor session 1 type erspan-source
source interface Gi0/0/0 rx
filter access-group ACL120
destination
erspan-id 1
ip address 20.1.1.2 <== Ip address of L3 interface on APP
origin ip address 68.68.68.68
```

Configuration Examples for Application Hosting

See the following examples:

Example: Enabling IOx

```
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 secret 0 cisco
Device(config)# end
```

Example: Configuring a VirtualPortGroup to a Layer 3 Data Port

```
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
```

Example: Installing and Uninstalling Apps

```
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
```

Example: Overriding the App Resource Configuration

```
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
```

Signed Application Support

To install a signed application, signed verification has to be enabled on the device. Signed verification can be enabled or disabled by the following command:

# **app-hosting verification** {**enable**|**disable**}

The signed verification enabled or disabled status can be verified by the **show app-hosting infra** command:

```
# show app-hosting infra
IOX version: 2.7.0.0
App signature verification: disabled
Internal working directory: /vol/harddisk/iox

Application Interface Mapping
AppGigabitEthernet Port # Interface Name Port Type Bandwidth
1 AppGigabitEthernet0/1/1 KR Port - Internal 10G


CPU:
Quota: 99(Percentage)
Available: 99(Percentage)
Quota: 3465(Units)
Available: 0(Units)
```

When signed verification is enabled, any unsigned app can not be activated, and signed app can move to different states irrespective of the app sign verification enabled or disabled.

After enabling the signed verification, follow the instructions in Installing and Uninstalling Apps, on page 10 to install the application.

Cisco Cyber Vision

Cisco Cyber Vision Center (CVC) gives more visibility into Industrial IoT networks across Industrial Control Systems (ICS) with real-time monitoring of control and data networks. On IoT IOS-XE platforms beginning with release 17.4, integration of CVC is supported by deploying IOX Cyber Vision sensor. With this sensor deployed on IoT Routers, the platform can forward the traffic from IOX applications to Cyber Vision Center for real-time monitoring and we can forward any captured PCAP files to Vision center from IOX application. The minimum Cyber Vision release is 4.1.1 to work with the IR8340. For more information about CVC, see the release notes in the following URL:

https://www.cisco.com/c/en/us/support/security/cyber-vision/products-release-notes-list.html

For more information about CVC installation and ERSPAN with CVC, see the following:

https://www.cisco.com/c/en/us/td/docs/routers/access/1101/software/configuration/guide/b_IR1101config/m-new-features-17-4-1.html