

IOx Application Hosting

This chapter contains the following sections:

- Information About Application Hosting, on page 1
- Application Hosting on the IR1101 Router, on page 4
- How to Configure Application Hosting, on page 8
- Installing and Uninstalling Apps, on page 14
- Overriding the App Resource Configuration, on page 15
- Verifying the Application Hosting Configuration, on page 16
- Digital IO Enhancement, on page 18
- Configuration Examples for Application Hosting, on page 18
- IOx Access to USB Storage, on page 19
- Cisco ThousandEyes Enterprise Agent, on page 20

Information About 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 module describes the Application Hosting feature and how to enable it.

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.

IOx architecture for the IR1101 is different compared to other Cisco platforms that use the hypervisor approach. In other platforms, IOx runs as a virtual machine. IOx is running as a process on the IR1101.

Cisco Application Hosting Overview

The IR1101 enables the user to deploy the application using the app-hosting CLIs. These app-hosting CLIs are not available on the other older platforms. There are additional ways to deploy the applications using the Local Manager and Fog Director.

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 access to services via REST APIs.
- 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.

GPS access to IOx Apps

Previously, when a modem has GPS enabled, the NMEA stream was not forwarded to IOx. This release allows the NMEA stream to be forwarded to IOx from the ngiolite module. There are two steps to enable this.

- Create a tunnel between Linux and IOx
- Forward all NMEA messages over the tunnel to IOx.

The system code checks for the presence of the tunnel, and if it is not present, data cannot be sent to IOx.

To support this feature there will be two new tunnels created for two cellular modems on the IR1101 and IR1800. Two tunnels are created by default and whichever modem has the GPS/NMEA enabled, the NMEA stream will be sent over the corresponding tunnel as follows:

Modem0:

[Linux] /dev/ttyTun5 and /dev/ttyTun6 [IOx]. Soft link to /dev/ttyTun5 will be created named /dev/ttyTunNMEA0, soft link to /dev/ttyTun6 will be created named /dev/ttyNMEA0 which can be accessed from IOx.

Modem1:

[Linux] /dev/ttyTun7 and /dev/ttyTun8 [IOx]. Soft link to /dev/ttyTun7 will be created named /dev/ttyTunNMEA1, soft link to /dev/ttyTun8 will be created named /dev/ttyNMEA1 which can be accessed from IOx.

The following command shows the state of the GPS:

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

Guest Shell as IOx Container APP

The Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. Using the Guest Shell, the user can also install, update, and operate third-party Linux applications and access the IOS CLI.

The Guest Shell environment is intended for tools, Linux utilities, and manageability rather than networking.

Guest Shell shares the kernel with the host (router) system. Users can access the Linux shell of Guest Shell and update scripts and software packages in the container rootfs. However, users within the Guest Shell cannot modify the host file system and processes.

The Guest Shell container is managed using IOx. IOx is Cisco's Application Hosting Infrastructure for Cisco IOS XE devices. IOx enables hosting of applications and services developed by Cisco, partners, and third-party developers in network edge devices, seamlessly across diverse and disparate hardware platforms.

The Guest Shell is typically bundled with the system image and can be installed using the **guestshell enable** Cisco IOS command. However, this approach leads to an increase of roughly 75MB in the size of the image. This is a problem for some users who have limited bandwidth, or download images through LTE.

With these users in mind, guestshell will be made available as a single tar file which can then be downloaded and installed on the system like any other IOX application. As a result, there won't be any increase in the size of the universal release image.



Note Day 0 guestshell provisioning will not work with this approach.

By default, Guest Shell allows applications to access the management network via the management interface. For platforms like the IR1101, which don't have a dedicated management port, a VirtualPortGroup can be associated with Guest Shell in the IOS configuration.

Sample guestshell configuration can be found here.

To install guestshell on the device, copy the tar file to the router and run the following command:

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

Use the following command to check the status:

```
show app-hosting list
```

Once guestshell has been deployed successfully, standard guestshell commands such as **guestshell enable**, **guestshell run bash**, and **guestshell run python3** should work.

The following resource talks about running python scripts using guestshell:

CLI Python Module

Note Only python3 is supported in 17.5.1.

Important - Before You Install

Before attempting to install Guest shell on your device, please verify that the device has IOx container keys programmed on it by running the following command:

```
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
```

The output should contain one or more lines with the Product Name “Cisco Services Containers”. If the device doesn’t have container keys programmed on it, then you won’t be able to install guest shell.

You will see an error like the following:

```
*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.
```

There is no software based mechanism to install container keys on the device. The keys have to be programmed at the manufacturing facility. IR1100 devices shipped after January 1, 2020, should have the container keys programmed.

The guest shell tar file is published along with the IOS-XE image for a given release. More information can be found here:https://developer.cisco.com/docs/iox/#!iox-resource-downloads/downloads

Application Hosting on the IR1101 Router

This section describes the application-hosting characteristics specific to the IR1101 Industrial Router.

Note

- The IR1101 CPU is not based on x86 architecture like other Routers. Therefore, this requires the application to comply with the ARM 64-bits architecture.
- Before installing or activating an IOx application from Local Manager, you need to configure the required Virtual Port Group interfaces.

Application hosting can be achieved using the app-hosting cli's as well using the Local Manager and Fog Director.

IOx URL Access Methods

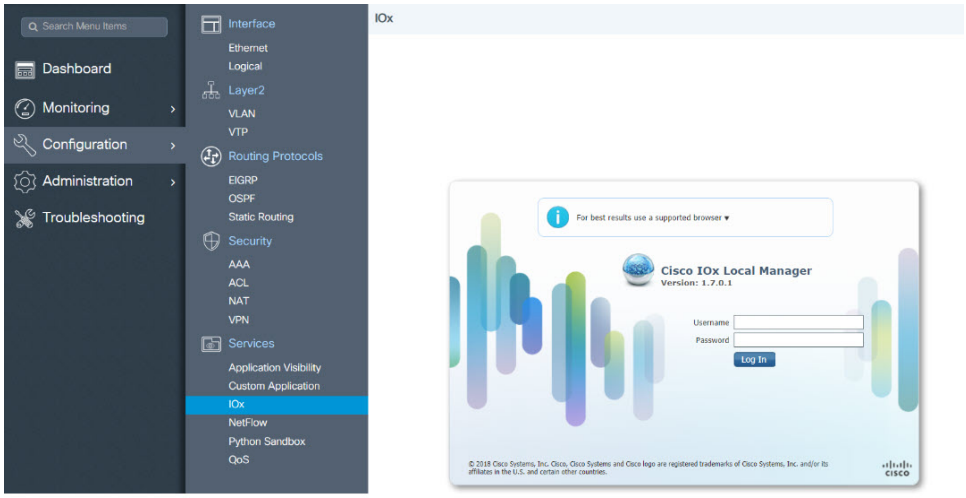
The IOx URL can be accessed in two different ways.

1. Using the direct URL to the IOx login.
2. Navigate to the IOx login through the Web User Interface (WebUI)

The syntax for the first method is **https://***IR1101-IP-ADDRESS***/iox/login**

The syntax for the second method is **https://***IR1101-IP-ADDRESS* and then navigate to IOx as shown in the following:

Figure 1: Local Manager



1. From the WebUI, click on **Configuration > Services > IOx**
2. Login using the username and password configured.
3. Follow the steps for the application life-cycle in the Cisco IOx Local Manager Reference Guide.

IOX URL User Restriction

The second method will make the configuration of the entire router available to IOx users. In some organizations, the IOx users are different from those that manage and administer the router. In this case, there is a need for restricting the access of the IOx users to ONLY the IOx Local Manager WebUI and not the entire WebUI of the router.

Currently, IOx users are configured as privilege 15 users. To restrict the IOx users to ONLY the Local Manager, the following commands can be used:

```
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
```

The command **no ip http server** will turn off the web server without https. The next command **ip http secure-server** is to turn on the https mode.

If you include only **OPENRESTY_PKI AND NG_WEBUI**, then you will be enabling ONLY the IOX local manager modules, and hence ALL users can ONLY access the IOX local manager if they have privilege 15, https://IR1101-IP-ADDRESS/iox/login.

And for ALL user, the WebUI access , **https://***IR1101-IP-ADDRESS* will be disabled.

Note This method will disable the main web page **https://***IR1101-IP-ADDRESS* for all users and will enable only **https://***IR1101-IP-ADDRESS***/iox/login** for all users. Use this method if you do not use the IR1101 main router WebUI for general administration and configuration.

VirtualPortGroup

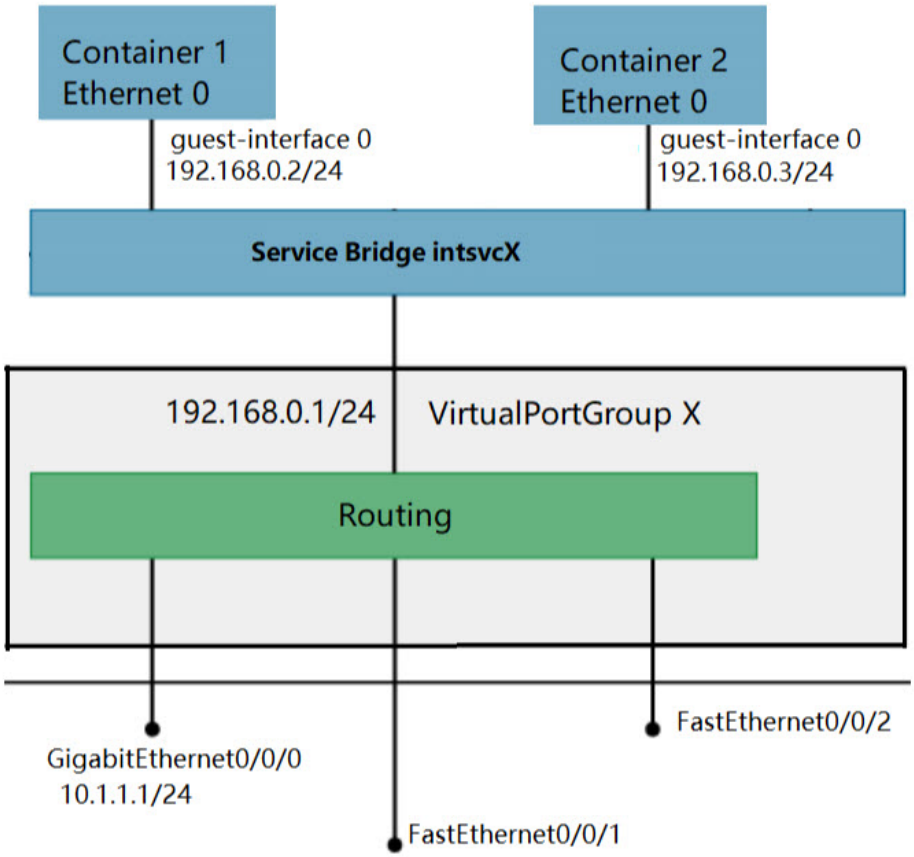
The VirtualPortGroup is a software construct on Cisco IOS that maps to a Linux bridge IP address. As such, the VirtualPortGroup represents the switch virtual interface (SVI) of the Linux container. Each bridge can contain multiple interfaces; each mapping to a different container. Each container can 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, as it is different than the IR8x9 routers.

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

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.

DETAILED STEPS

Steps	Command	Purpose
1.	**enable** Example: `Device>enable`	Enables pr EXEC mo Enter your password i prompted.
2.	**configure terminal** Example: `Device#configure terminal`	Enters glo configurat mode.
3.	**iox** Example: `Device(config)#iox`	Enables IO
4.	**ip http server** Example: `Device(config)#ip http server`	Enables th server on y or IPv6 sy
5.	**ip http secure-server** Example: `Device(config)#ip http secure-server`	Enables a HTTP (HT server.

Steps	Command	Purpos
6.	**username** *name* **privilege** *level* **password** {**0** | **7** | *user-password* }*encrypted-password* Example: `Device(config)#username cisco privilege 15 password 0 cisco`	Establi usernar authen system privileg the use The us privileg must be as 15.
7.	**end** Example: `Device(config-if)#end`	Exits in configu mode a to priv EXEC

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.

DETAILED STEPS

Step	Command
1.	**enable** Example: `Device>enable`
2.	**configure terminal** Example: `Device#configure terminal`
3.	**ip routing** Example: `Device(config)#ip routing`

Step	Command
4.	**interface type number** Example: `Device(config)#interface gigabitethernet 0/0/0`
5.	**no switchport** Example: `Device(config-if)#no switchport`
6.	**ip address ip-address mask** Example: `Device(config-if)#ip address 10.1.1.1 255.255.255.0`
7.	**exit** Example: `Device(config-if)#exit`
8.	**interface type number** Example: `Device(config)#interface virtualportgroup 0`
9.	**ip address ip-address mask** Example: `Device(config-if)#ip address 192.168.0.1 255.255.255.0`
10.	**end** Example: `Device(config-if)#end`
11.	**configure terminal** Enter configuration commands, one per line. End with CNTL/Z. Example: `Device#configure terminal`

Step	Command
12.	**app-hosting appid app1** Example: `Device(config)#app-hosting appid app1`
13.	**app-vnic gateway0 virtualportgroup 0 guest-interface 0** Example: `Device(config-app-hosting)#app-vnic gateway0 virtualportgroup 0 guest-interface 0`
14.	**guest-ipaddress 192.168.0.2 netmask 255.255.255.0** Example: `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** Example: `Device(config-app-hosting-gateway0)#app-default-gateway 192.168.0.1 guest-interface 0`
16.	**end** Example: `Device#end`

Native docker support

Native Docker Support has been added to the 17.2.1 release. This feature enables users to deploy the docker applications on the IR1101. The application lifecycle process is similar to the procedure in the Installing and Uninstalling Apps section. For docker applications, entry point configuration is required as part of the application configuration. Please refer to the following example for the entry point configuration.

```
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#
```

The output for docker applications is shown in the following example:

```
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#
```

Digital IO for IOx container applications

Release 17.2.1 provides support for IOx container applications to be able to access the digital IO. There is a new CLI that has been added to the alarm contact command.

```
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
```

Enabling the **attach-to-iox** command will provide complete control of all Digital IO ports to IOx. The ports will be exposed as four character devices /dev/dio-[1-4] to IOX applications. You can use read/write functions to get/set values of the Digital IO ports.

If you wish to update the mode, you can write the mode value to the character device file. This is accomplished by IOCTL calls to read/write the state, change mode, and read the true analog voltage of the port. Following this method, you can attach analog sensors to the IR1101. All ports are initially set to Input mode with voltage pulled up to 3.3v.

The following are examples of IOCTL calls:

Read Digital IO Port

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

Write to Digital IO Port

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

Change mode

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

List of IOCTLs supported

```
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
```

Read State using IOCTL

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

Change mode using IOCTL

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

Signed application support

Cisco signed applications are now supported on the IR1101 router. To install a signed application, signed verification has to be enabled on the device. Signed verification can be enabled using the **app-hosting verification enable** command.



Note Third-party application signing and the installation of third-party signed apps are currently not supported or available on this platform.

```
Router#app-hosting verification enable
```

After enabling the signed verification, you can install the application.

Installing and Uninstalling Apps

DETAILED STEPS

Step	Command
1.	**enable** Example: `Device>enable`
2.	**app-hosting install appid** *application-name* **package** *package-path* Example: `Device#app-hosting install appid lxc_app package flash:my_iox_app.tar`
3.	**app-hosting activate appid** *application-name* Example: `Device#app-hosting activate appid app1`
4.	**app-hosting start appid** *application-name* Example: `Device#app-hosting start appid app1`
5.	**app-hosting stop appid** *application-name* Example: `Device#app-hosting stop appid app1`

Step	Command
6.	**app-hosting deactivate appid** *application-name* Example: `Device#app-hosting deactivate appid app1`
7.	**app-hosting uninstall appid** *application-name* Example: `Device#app-hosting uninstall appid app1`

Overriding the App Resource Configuration

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

DETAILED STEPS

Step	Command
1.	**enable** Example: `Device>enable`
2.	**configure terminal** Example: `Device#configure terminal`
3.	**app-hosting appid** *name* Example: `Device(config)#app-hosting appid app1`

Step	Command
4.	**app-resource profile** *name* Example: `Device(config-app-hosting)#app-resource profile custom`
5.	**cpu** *unit* Example: `Device(config-app-resource-profile-custom)# cpu 800`
6.	**memory** *memory* Example: `Device(config-app-resource-profile-custom)# memory 512`
7.	**vcpu** *number* Example: `Device(config-app-resource-profile-custom)# vcpu 2`
8.	**end** Example: `Device(config-app-resource-profile-custom)# end`

Verifying the Application Hosting Configuration

DETAILED STEPS

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) 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

Displays detailed information about the application.

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

Displays the list of applications and their status.

Example:

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

Digital IO Enhancement

Support has been added to allow some digital I/O ports to be managed by IOSd, and some other digital IO ports to be managed by IOx container apps. An updated CLI has been added and the YANG model for Digital IO Enhancement has been updated.

The 17.5.1 version of the CLI is:

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



Note With release 17.5.1, **alarm contact attach-to-iox** gave IOX control for ALL digital IO ports (1 thru 4).

The 17.6.1 version of the CLI is:

```
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.
```

In the updated CLI, <1-4> are the number of digital I/O ports to assign to IOx for container apps.



Note With release 17.6.1, each digital IO port can be assigned to IOX individually.

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 password 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
```

IOx Access to USB Storage

Customers have requested the ability to mount the host a USB thumb drive within the Docker container running on IOx. The bootflash has a limited number of read/write cycles, and a container continuously writing on the eMMC would prematurely wear out the unit. Using the USB thumb drive will allow Docker containers to write in a continuous manner without compromising the integrity of the bootflash.

Feature Requirements and Limitations

The following apply to this feature:

- The filesystem types supported for USB thumb drives on the IR1101 are: VFAT, EXT2 and EXT3. However, IOx only supports mounting of USB thumb drives with EXT2 and EXT3 filesystem. Cisco recommends EXT3 for the following reasons:
  - EXT3 is a journaling filesystem, which means there are not fragmentation issues.
  - Read/Writes are significantly faster with EXT3 filesystems
  - VFAT has a 4 GB maximum file-size limitation, which is a problem with container continuously writing large files.
- If the USB thumb drive is removed while a write operation by IOx apps is in progress, all the files included in the copy operation will be lost.
- If the USB thumb drive is removed while IOX and the app are using it, IOX will still be in running state. The functionality of the app using USB thumb drive as storage will be severely impacted, since it will not be able to read and/or write on the USB thumb drive.

Making the USB Thumb Drive Available to the IOx App

In order to make the USB thumb drive available to the IOx app, you need to issue a run option. See the following example:

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

This command will mount the USB thumb drive file system within the IOx application filesystem, and it will be available in the /usbflash0 folder, as showed by the following log from an IOx application:

```
/ # 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
#
```

Cisco ThousandEyes Enterprise Agent

A Cisco ThousandEyes Enterprise Agent is a network monitoring solution that

- runs a variety of tests using agents
- provides real-time monitoring of network and application performance, and
- offers multidimensional insights with routing and device data for end-to-end visibility.

Additional information

The Cisco ThousandEyes Enterprise Agent enables you to view end-to-end paths across networks and services affecting your business. It actively monitors network traffic paths across internal, external, and internet networks, helping analyze network performance and application availability.

You can use application-hosting features to deploy the Cisco ThousandEyes Enterprise Agent as a container application on Cisco Industrial IoT Routers. The agent runs as a Docker container using the Cisco IOx.

Starting with Cisco IOS XE Release 17.18.2, you can configure Cisco ThousandEyes Enterprise Agent in Controller mode.

For details about configuring Cisco ThousandEyes Enterprise Agent in controller mode, see Cisco SD-WAN Systems and Interfaces Configuration Guide.

For more information about Cisco ThousandEyes Enterprise Agent, see ThousandEyes Documentation.