IOx Components Configuration Guide

Kinetic - Edge & Fog Processing Module (EFM) 1.6.0

Revised: December 11, 2018

Table of Contents

Introduction

The provided EFM IOx application images include the EFM C Broker so that they can be deployed on a Cisco IOx platform with an x86 architecture (such as the Cisco IR809, IR829 and IC3K) or the PowerPC architecture (such as the Cisco IE 4000).

For the IR809/IR829/IC3K, the image is provided with application development languages (DART or java) that the microservices may need to execute (e.g., dart-system or java-snmp). Other components, such as the file system, have been added to the images to allow storage of data during operation.

IOx allows each application to have its own life cycle management in order to enable upgrading and restarting these applications independently. In the EFM system, the EFM Link Manager provides life cycle management to EFM microservices in the same application container. Using this facility, life cycle management of EFM microservices can be consistent with the management of other nodes throughout the EFM system using the EFM System Administrator.

Due to the memory and disk space constraints of the IOx target platforms, care has been taken to minimize the installation of too many applications languages support (such as Java, DART, and python) as well as microservice links.

The “ioxclient” and Local Manager (a graphical tool) can deploy the Cisco IOx EFM applications.

Once an application is deployed into an IOx host and the application is running, it is possible to use the EFM System Administrator tool, currently only supported on a Linux environment, to configure the new message broker and links by creating a new uplink connection to the router’s outside address of the application (see NAT statements below).

The C++ Broker is a multi-threaded high-performance broker with very low footprint in order to leverage the multi-core capability of different platforms. This allows allow for performance on the edge. The C++ broker scales and performs so well that we recommend to use it on all other levels, except when UI is required, in a multi-tier architecture.

The EFM-IOx-1-6-0.zip package provide the following packages and subfolders:

For the Cisco IE 4000 platform:

- efm_ppc: LXC Package suitable for installation onto an ie4k router running an IOx environment. Components running on the Cisco IE 4000 device:
    - IOT-DSA broker (C++ based)
    - Life cycle manager

For the Cisco IR8x9 and IC3K platform:

- efm_x86_64: LXC Package suitable for installation onto an IR809/IR829/IC3K router running an IOx environment.
    - IOT-DSA broker (C++ based)
    - Lifecycle manager
    - DART runtime environment
    - JAVA runtime environment
    - dslink-dart-system link
    - dslink-dart-dql link
    - dslink-dart-dataflow engine

More information about IOx can be found at Cisco's DevNet site: https://developer.cisco.com/site/iox/index.gsp .

IOx-supported version

The current version supported for the IR809/IR829 is IOx version 1.4. For the CGR1K, it is IOx version 1.2.3. On Cisco IE 4000, it is IOx version 1.3.

Upward compatibility to newer IOx versions was not tested with EFM 1.6.0.

EFM IOx application profile

activate.json file

The EFM Application profile file activate.json defines the resources and names that are used to the IOx guest OS environment. This file can be used for the default activation phase of the IOx deployment, after the installation of the corresponding package.tar.gz.

The default activation profile will be "large". This is defined as:

CPU (cpu-units): 60

Memory (MB): 256

Disk (MB): default

In many cases, the most memory is recommended and will require a manual definition of the custom profile described in the next section.

Defining and activating the custom profile

Due to changes in the IOx version 1.4, it may be best to deploy with a custom profile rather than with the "activate.json" profile. This will allow reserving all of the available application memory beyond the default value when not installing and running other IOx applications in parallel.

For example, on the IR809/829 running IOx version 1.4, a custom profile can reserve the maximum values for the EFM:

- CPU (cpu-units): 732
- Memory (MB): 767
- Disk (MB): 256

Disabling IOx application package signature verification for installation

IOx 1.4 introduces the concept of package signature verification. The EFM IOx application is not self-signed and does not install with the default verification enable[1]. Installation will require using the ioxclient to disable the verification. Enter the following:

```
ioxclient platform signedpackages disable
```

Accessing IR8x9 serial ports and gyroscope

The EFM 8x9 IOx package allows for the reservation and communication with three serial interfaces. On the IR829, this can correspond to the two serial interfaces on the router and the gyroscope/accelerometer. For IR809, the additional serial interface is redundant.

The activate.json profile file maps **does not** map the serial interfaces in a predefined manner.

In order to map the serial interfaces to the logical Serial Adaptors, the Local Manager tool **must be used**.

Using IOx-NAT or IOx-Bridging for the EFM application

The EFM application is installed on the IOx Guest OS. To network outside the router or switch, it is necessary to understand how the EFM application obtains its IP address and exposes to the rest of the network.

For IPv4, the EFM application obtains its address using a DHCP request. The application obtains its address in two different ways, depending on the mode of configuration of eth0 and/or eth1.

Network Configuration of eth0/eth1	Source of IP Address	Notes:

[1] See https://developer.cisco.com/site/iox/docs/#manage-package-signature-validation for more details.

IOx-nat	The IPv4 DHCP address is obtained from an INTERNAL pool inside the IOx GuestOS.	• All connectivity from the application is NAT'ed via the GuestOS IP address on the router or switch. • Start and restart events do not affect the router NAT statements to reach the GuestOS. • The application internal TCP port is mapped to a free GuestOS TCP port (for example, the efm ports 8080 and 8484 are custom mapped to the IOx GuestOS 8080 and 8484).
IOx-bridge	The IPv4 DHCP address is obtained from an EXTERNAL pool outside the IOx GuestOS. All communications flow around the GuestOS, directly to the application.	• No TCP port mapping occurs and the application TCP ports are exposed according to the application profile. For the EFM, TCP ports 8080 for http and 8484 for https • Every start and restart of the application requests a new IPv4 address. In many cases, this can affect the NAT statements on the router. • The router or switch may not have visibility into the IP address assigned and this may cause challenges in obtaining the address for connecting to the device.

While either mode can be used, this document will guide you with using iox-nat because it is simpler to use to determine the IPv4 address that is needed to connect the upstream broker.

Mapping the EFM application TCP port

The EFM application package has defined a custom mapping of the TCP Ports 8080 and 8484. These TCP ports will be exposed either using iox-nat or iox-bridge. If these values overlap with another application, the custom port mapping values can be modified via the Local Manager deployment interface.

While Port 8484 is exposed, the broker does not listen to https connections until the server certificate and key file are installed and the broker.json file is properly updated.

Deploying with the IOx Client

IOx client is supported on Windows, Mac, and Linux. To obtain the latest version of ioxclient, see https://developer.cisco.com/site/iox/docs/#none-downloads.

Using the IOx client to install the EFM application on the IOx platform:

```
cd <EFM package name folder>
ioxclient app install <IOx app name> package.tar
```

Example:

```
#ioxclient app install EFM package.tar
Currently active profile :  default
Command Name: application-install
Installation Successful. App is available at :
https://192.168.25.201:8443/iox/api/v2/hosting/apps/EFM_Broker
Successfully deployed
```

Starting the application with IOx client

```
ioxclient app activate <IOx app name> --payload activate.json
```

Example:

```
#ioxclient app activate EFM --payload activate.json
Currently active profile :  default
Command Name: application-activate
Payload file : activate.json. Will pass it as application/json in request body..
App EFM_Broker is Activated
```

```
ioxclient app start <IOx app name>
```

Example:

```
#ioxclient app start EFM
Currently active profile :  default
Command Name: application-start
App EFM_Broker is Started
```

Verifying that the application is running with IOx client

```
ioxclient app list

#ioxclient app list
```

```
Currently active profile :  default
Command Name: application-list
List of installed App :
 1. EFM        --->    RUNNING
```

Determining the GuestOS IP address on the IR809/IR829/IC3K

To determine the application container IP address for the NAT mapping on the router:

```
show iox host list detail

ir829#show iox host list detail

IOX Server is running. Process ID: 331
Count of hosts registered: 1

Host registered:
===============
    IOX Server Address: FE80::235:1AFF:FE91:FA8C; Port: 22222

    Link Local Address of Host: FE80::1FF:FE90:8B05
    IPV4 Address of Host:       192.168.101.6
    IPV6 Address of Host:       fe80::1ff:fe90:8b05
    Client Version:             0.4
    Session ID:                 1
    OS Nodename:                ir829-GOS-1
    Host Hardware Vendor:       Cisco Systems, Inc.
    Host Hardware Version:      1.0
    Host Card Type:             not implemented
    Host OS Version:            1.5.5.1
    OS status:                  RUNNING

    Interface Hardware Vendor:  None
    Interface Hardware Version: None
    Interface Card Type:        None
```

In the example above, the GuestOS is 192.168.101.6.

Detailed Steps:

The example assumes the following:

- The router or switch has been pre-configured for networking access
- The IOx Guest OS is network reachable from the remote computer that will execute ioxclient
- a predefined IOx Guest profile for the IOx GuestOS

Create an ioxclient profile for the IOx GuestOS host

	Command or Action	Purpose

Step 1	**`$ ioxclient profiles create`**
	
	`Active Profile :  default`
		Install the EFM application package.tar with the name efm

Step 2	`Enter a name for this profile :` **`default`**	Activate the EFM application using the default settings in the activate.json file
Step 3	`Your IOx platform's IP address[127.0.0.1] :` **`192.168.25.201`**	Start the EFM application
Step 4	`Your IOx platform's port number[8443] :`	Type ENTER for default
Step 5	`Authorized user name[root] : root`	Type administrator user defined in the IOS configuration. Assuming root
Step 6	`Password for root :`	Type administrator password defined in the IOS configuration.
Step 7	`Local repository path on IOx platform[/software/downloads]:`	Type ENTER for default
Step 8	`URL Scheme (http/https) [https]:`	Type ENTER for default
Step 9	`API Prefix[/iox/api/v2/hosting/]:`	Type ENTER for default
Step 10	`Your IOx platform's SSH Port[2222]:`	Type ENTER for default
Step 11	`Your RSA key, for signing packages, in PEM format[]:`	Type ENTER for default

Step 12	`Your x.509 certificate in PEM format[]:`
	`Activating Profile  default`
	`Saving current configuration`
		Type ENTER for default

Installing, activating and starting the EFM IOx Application

	Command or Action	Purpose

	`$ ioxclient platform signedpackages disable`	Disable package signature validation on the platform

Step 1	`$ ioxclient app install EFM package.tar`
	
	`Currently active profile :  default`
	`Command Name: application-install`
	`Installation Successful. App is available at :`
	`https://192.168.25.201:8443/iox/api/v2/hosting/apps/EFM_Broke`
	`r`
	`Successfully deployed`
		Install the EFM application package.tar with the name EFM

Step 2	`$ ioxclient app activate EFM --payload activate.json`
	
	`Currently active profile :  default`
	`Command Name: application-activate`
	`Payload file : activate.json. Will pass it as`
	`application/json in request body..`
	`App EFM Broker is Activated`
		Activate the EFM application using the default settings in the activate.json file

Step 3	`$ioxclient app start EFM`
	
	`Currently active profile :  default`
	`Command Name: application-start`
	`App EFM_Broker is Started`
		Start the EFM application

Applying NAT to IR809/IR829 router configuration (if needed)

If global routing reachability is not available for the subnet belonging to the GuestOS, then inserting an IP Network Address Translation (NAT) can allow other brokers to reach the GuestOS-hosted EFM broker.

Note: This NAT function is independent of the GuestOS internal NAT operation for applications, this NAT function is performed on the IOS Router to allow for global networking reachability beyond the IOx host.

For example, assume Vlan1 is the external address as shown below for the example above. The GuestOS exposes the ports 8080 and 8484 for http and https. They are going to be mapped externally to ports 8080 and 8484:

```
interface Vlan1
 ip address 192.168.25.201 255.255.254.0
 ip nat outside

interface GigabitEthernet5
```

```
 ip address 192.168.101.1 255.255.255.0
 ip nat inside


ip nat inside source static tcp 192.168.101.6 8080 interface Vlan1 8080
ip nat inside source static tcp 192.168.101.6 8484 interface Vlan1 8484
```

Defining the GuestOS IP address on the Cisco IE 4000

On the GuestOS IP, address is defined in the switch configuration under “iox”.

For example:

```
iox
 host ip address 10.228.219.200 255.255.255.128 vlan 1
 host ip default-gateway 10.228.219.129
```

Obtaining the Application TCP port mapping with IOx client on the Cisco IE 4000

```
ioxclient app info <IOx app name>
```

The last step will help determine which IOx guest operating TCP ports have been assigned to the EFM Broker.

For example, using the ioxclient app info <app> to obtain the application information:

```
# ioxclient app info EFM
Currently active profile :  default
Command Name: application-info
Details of App : EFM
-----------------------------
{
 "appCustomOptions": "",
 "appType": "lxc",
 "author": "femasche",
 "authorLink": "mailto://femasche@cisco.com",
 "dependsOn": {},
 "description": "EFM ",
 "env": {
  "CAF_APP_APPDATA_DIR": "/data/appdata",
  "CAF_APP_CONFIG_DIR": "/data",
  "CAF_APP_CORE_DIR": "/local/local1/core_dir",
  "CAF_APP_CPU_SHARES": 5797,
  "CAF_APP_ID": "efm",
  "CAF_APP_LOG_DIR": "/data/logs",
  "CAF_APP_MEMORY_SIZE_KB": 262144,
  "CAF_APP_PERSISTENT_DIR": "/data",
  "CAF_APP_PERSISTENT_DISK_SIZE_KB": 10240,
  "CAF_APP_USERNAME": "root",
  "CAF_SYSTEM_UUID": "e91e78ba-4065-43ff-a9a7-86339fe0eb20"
 },
 "id": "efm",
 "is_autoinstalled": false,
 "is_service": false,
 "name": "efm",
 "networkInfo": {},
 "resources": {
  "cpu": 600,
  "disk": 10,
  "memory": 256,
  "network": [
   {
    "interface-name": "eth0",
    "network-name": "iox-nat0",
    "port_map": null,
    "ports": {
     "tcp": [
      8080,
      8484
     ]
    }
   }
  ],
  "profile": "c1.large",
  "vcpu": 1
 },
 "state": "RUNNING",
 "toolkitServicesUsed": null,
 "version": "1.20"
}
```

In the example above, the section “ports” shows that the application-defined TCP Port 8080 and TCP port 8484 are available. In the EFM instance, Port 8080 is for http and Port 8484 for https connections.

Configuring the CGR1000 network interfaces

For CGR1000 devices, we need to use the bridge mode for the containers' interfaces (iox-bridge). In bridge mode, the EFM application container might receive different IP addresses on every start. We need to define a specific IP address for the **IOS** interface to provide the application container with a static DHCP address and a forwarding rule for the EFM ports to the container. On the CGR1000, the following configuration defines static client-identifier:

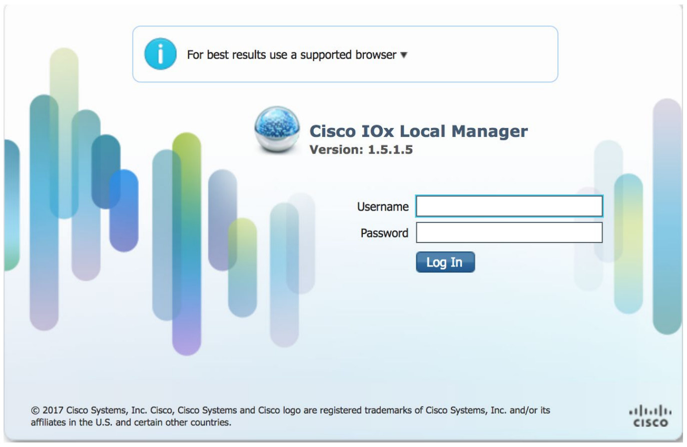
```
ip dhcp pool iox-efm-eth0-static
 host 192.168.4.2 255.255.255.0
 client-identifier 6566.6d
 default-router 192.168.4.1

ip dhcp pool iox-efm-eth1-static
 host 192.168.4.3 255.255.255.0
 client-identifier 6566.6d32
 default-router 192.168.4.1

ip nat inside source static tcp 192.168.4.2 8080 interface GigabitEthernet2/2 8080
```

The client identifiers are patched into the EFM application container. If you change the client identifiers in the IOS rules, you have to change the addresses provided to the udhcpc_opts commands inside the EFM application container in /etc/network/interfaces accordingly.

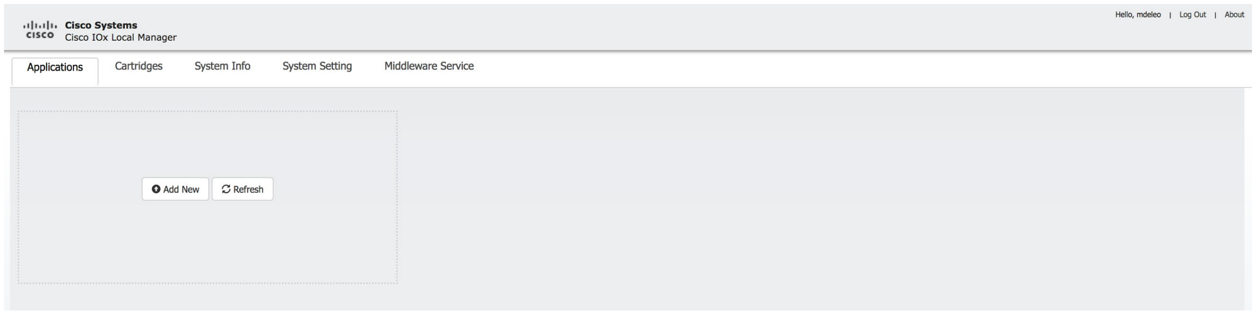
Deploying with Cisco IOx Local Manager

Caveat: When working with the IOx Local Manager, the browser language must be set to English. If not, a blank page will display.

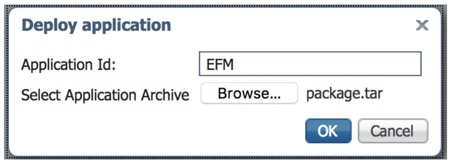
1. Connect via a web browser to the IOx router to the defined port for the Cisco IOx Local Manager. For example: https://192.168.25.201:8443/. Log in with the router credentials.


For best results use a supported browser ▾
Cisco IOx Local Manager
Version: 1.5.1.5
Username
Password
Log In
© 2017 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.

2. Click **Add New** to install the EFM application package.

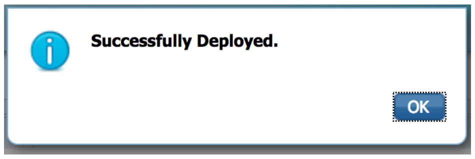


3. Apply any name for the application on the host, for example “EFM”. Locate the package.tar you are installing specific for the platform on your local disk. Then click **OK**. Upload will take a few minutes.

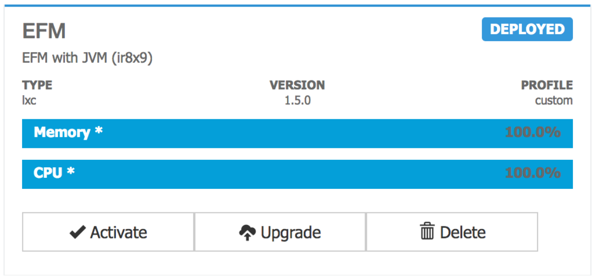

Deploy application

Application Id: EFM

Select Application Archive: Browse… package.tar


Successfully Deployed.

The following application status should display:


EFM

DEPLOYED

EFM with JVM (ir8x9)

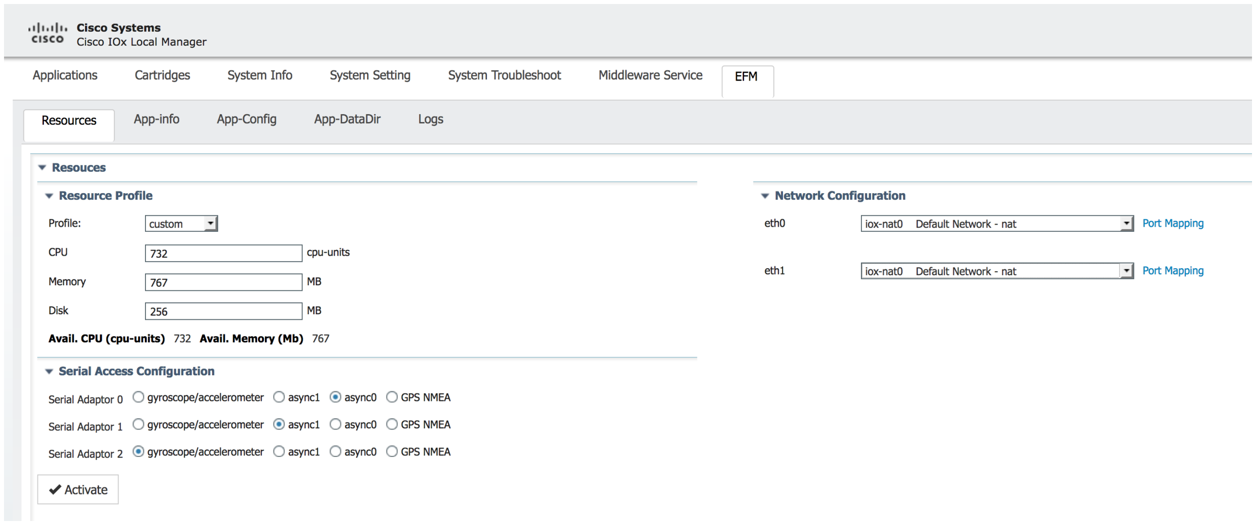
TYPE	VERSION	PROFILE
lxc	1.5.0	custom

Memory * 100.0%

CPU * 100.0%

4. Click **Activate** and the following page will appear to activate the EFM application. Under the **Resources** tab and the **Resources Profile**, the default values are shown.

The maximum available CPU and memory are shown at the bottom of the page. If needed, the values can be adjusted higher or lower as CPU and memory may vary if other applications are deployed.



Profile:	custom	
CPU	732	cpu-units
Memory	767	MB
Disk	256	MB

Avail. CPU (cpu-units) 732 **Avail. Memory (Mb)** 767

eth0	iox-nat0 Default Network - nat
eth1	iox-nat0 Default Network - nat

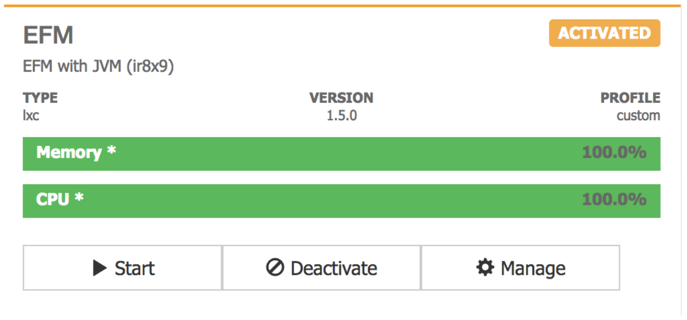
Serial Adaptor 0	gyroscope/accelerometer	async1	[✓ async0]	GPS NMEA
Serial Adaptor 1	gyroscope/accelerometer	[✓ async1]	async0	GPS NMEA
Serial Adaptor 2	[✓ gyroscope/accelerometer]	async1	async0	GPS NMEA

5. Serial ports need to be defined to proceed with the activation, even if the application will not communicate with any serial devices. There are no default selections.

6. Ensure that the Network Configuration is set to “iox-nat0 Default Network – nat,” if using NAT. Port Mapping should be left as is for auto.

7. Click **Activate**.

8. Return to the **Applications** tab.



EFM

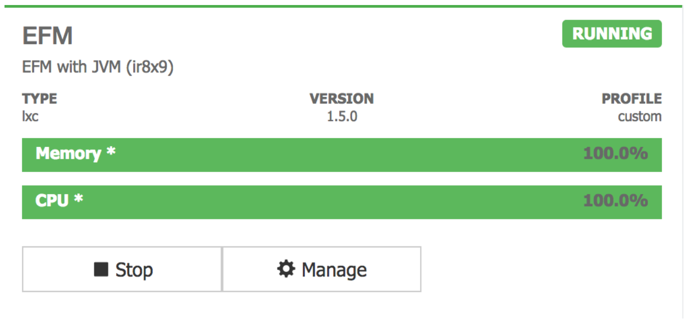
EFM with JVM (ir8x9)

TYPE	VERSION	PROFILE
lxc	1.5.0	custom

Memory * 100.0%

CPU * 100.0%

9. Now that application is activated, it must be started. Click **Start**.



Determining the GuestOS IP address on the IR809/IR829

To determine the application container IP address for the NAT mapping on the router:

```
show iox host list detail

ir829#show iox host list detail

IOX Server is running. Process ID: 331
Count of hosts registered: 1

Host registered:
===============
    IOX Server Address: FE80::235:1AFF:FE91:FA8C; Port: 22222

    Link Local Address of Host: FE80::1FF:FE90:8B05
    IPV4 Address of Host:       192.168.101.6
    IPV6 Address of Host:       fe80::1ff:fe90:8b05
    Client Version:             0.4
    Session ID:                 1
    OS Nodename:                ir829-GOS-1
    Host Hardware Vendor:       Cisco Systems, Inc.
    Host Hardware Version:      1.0
    Host Card Type:             not implemented
    Host OS Version:            1.4.2.3
    OS status:                  RUNNING

    Interface Hardware Vendor:  None
    Interface Hardware Version: None
    Interface Card Type:        None
```

In the example above, the GuestOS is 192.168.101.6.

Applying NAT to IR809/IR829 router configuration (if needed)

If global routing reachability is not available for the subnet belonging to the GuestOS, then inserting a IP Network Address Translation (NAT) can allow other brokers to reach the GuestOS-hosted EFM broker.

Note: This NAT function is independent of the GuestOS internal NAT operation for applications.

For example, assuming Vlan1 is the external address as shown below for the example above. The GuestOS exposes the Ports 8080 and 8484 for http and https. They are going to be mapped externally to Port 8080 and Port 8484:

```
interface Vlan1
 ip address 192.168.25.201 255.255.254.0
 ip nat outside

interface GigabitEthernet5
 ip address 192.168.101.1 255.255.255.0
 ip nat inside


ip nat inside source static tcp 192.168.101.6 8080 interface Vlan1 8080
ip nat inside source static tcp 192.168.101.6 8484 interface Vlan1 8484
```

Defining the GuestOS IP address on the Cisco IE 4000

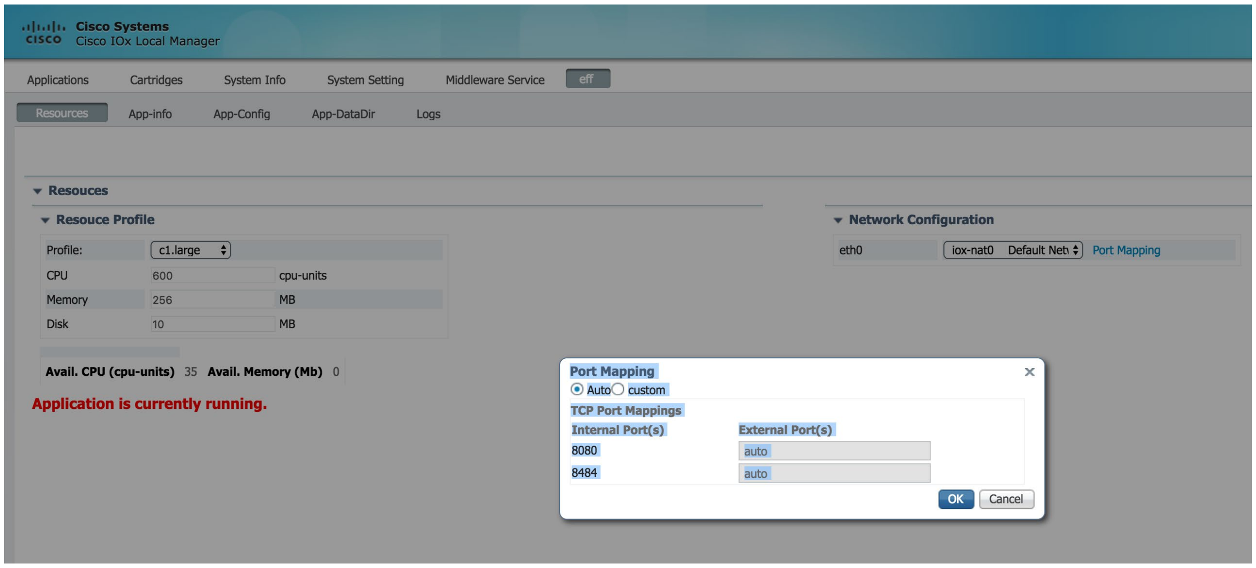
On the GuestOS IP, the address is defined in the switch configuration under **IOx**.

For example:

```
iox
 host ip address 10.228.219.200 255.255.255.128 vlan 1
 host ip default-gateway 10.228.219.129
```

Obtaining the Application TCP Port mapping with IOx client on the Cisco IE 4000

The figure below shows the port mapping in the Local Manager for the application under Resources.



Profile:	c1.large	
CPU	600	cpu-units
Memory	256	MB
Disk	10	MB

Avail. CPU (cpu-units) 35 Avail. Memory (Mb) 0

Application is currently running.

eth0 iox-nat0 Default Netv Port Mapping

Port Mapping
[✓ Auto] custom
TCP Port Mappings

Internal Port(s)	External Port(s)
8080	auto
8484	auto

In the example above, the section “ports” shows that the application-defined TCP Port 8080 and TCP port 8484 are available. These are automatically mapped externally to the IOx GuestOS address. In most cases, they are the same.

Configuring the CGR1000 network interfaces

For CGR1K devices, we need to use the bridge mode for the containers' interfaces (iox-bridge). In bridge mode, the EFM application container might receive different IP addresses on every start. We need to define a specific IP address in the **IOS** interface to provide the application container with a static DHCP address and a forwarding rule for the EFM ports to the container. On the CGR1000, the following configuration defines static client-identifier.

```
ip dhcp pool iox-efm-eth0-static
 host 192.168.4.2 255.255.255.0
 client-identifier 6566.6d
 default-router 192.168.4.1
```

```
ip dhcp pool iox-efm-eth1-static
 host 192.168.4.3 255.255.255.0
 client-identifier 6566.6d32
 default-router 192.168.4.1

ip nat inside source static tcp 192.168.4.2 8080 interface GigabitEthernet2/2 8080
```

The client identifiers are patched into the EFM application container. If you change the client identifiers in the IOS rules, you have to change the addresses provided to the udhcpc_opts commands inside the EFM application container in /etc/network/interfaces accordingly.

Caveats

Restarting the EFM application usually causes the application to request a new DHCP address. This will affect the NAT address statement and needs to be updated.

The DSLinks are stopped by default and need to be started, if required.

On CGR1K devices, files cannot be uploaded using the Local Manager due to a file permission problem. If you need to add an upstream to a CGR1K device, you have to copy the file into the application container using SCP or add the CGR to a broker as upstream and define everything remotely using the EFM System Administrator tool.

Configuring the EFM C++ Message Broker configuration files

The newly introduced EFM C++ message broker the C message broker option. The C++ message is meant as an option for users to instead the full EFM Server Dart message broker version by providing several benefits:

- Improved performance compared to DART Message Broker and C Broker
- Smaller memory footprint than DART Message Broker
- Feature compatibility between DART Message Broker and C++ Message Broker – except for the list from Felix/Lars, etc.
- Configuration consistency across all installation platforms (Linux/Windows/IOx)

The EFM C++ message broker allows for configuration of three different files rather than a single server.json file for the Dart broker. The system administrator can edit the text files. Modifications to this file should be performed when the broker is not running to avoid the content being overwritten by the message broker. The new configuration will take effect after startup.

Configuration files are located in the IOx app folder and does not necessarily contain all parameters:

The system administrator can edit the text files broker.json, manager.json and upstream.json. Modifications to these files should be performed when the broker is not running to avoid the content being overwritten by the message broker. The new configuration will take effect after startup.

broker.json example and parameters.

```
{
    "http": {
        "enabled": false,
        "port": 8080,
        "protocol": "dualstack"
    },
    "https": {
        "enabled": true,
        "port": 8443,
        "protocol": "dualstack",
        "certName": "server.pem",
        "certKeyName": "key.pem",
        "cert_chain_file": "server.ca-bundle",
        "tmp_dh_file": "dhparams.pem",
        "cipher_list": "HIGH:!aNULL"
    },
    "allowAllLinks": true,
    "workers": 1,
    "logging": {
        "log_level": "info",
        "debug_level": "no"
    },
    "ssl": {
        "self_signed_tls_certificate_allowed": true,
        "certs_path": "ca",
        "ca_file": "ca/ca-bundle.crt",
        "cipher_list": "HIGH:!aNULL",
        "verify_peer": true
    },
    "redo_log": {
        "path": ".redo",
        "max_entries_per_file": 1024,
        "max_size_per_file_bytes": 0,
        "max_files_per_log": 0,
        "flush_after_write": true,
        "automatic_recovery": true,
        "write_encrypted_values": true,
        "min_available_disk_space_threshold_mb": 50
    },
    "qos": {
        "default_queue_length": 1024
    },
    "max_send_queue_length": 8,
    "serializer": {
        "serialization_frequency": 1000,
        "serialize_values": true
    }
}
```

Section	Option	Default	Description
qos	default_queue_length	1024	Length of internal value queue
	max_send_queue_length	8	Specifies the maximum length of the internal send queue. If this number of send messages has not been acknowledged yet, the sending will be paused until at least some of these messages have been acknowledged. The default is 8.
serializer	serialization_frequency	1000	The node serialization will be called intermittently with this frequency in ms.
serializer	serialize_values	true	Controls if node values shall also be serialized. If set to false no values will be serialized.
redo_log	path	.redo	Path to the storage directory
redo_log	max_entries_per_file	1024	The maximum entries of each redo log file. The log will be cycled when this is reached. This limitation does not apply if set to 0.
redo_log	max_size_per_file_bytes	0	The maximum size (in bytes) of each redo log file.
redo_log	max_files_per_log	0	The maximum number of files in each redo log folder. The latest log will be deleted if this is reached. This limitation does not apply if set to 0.
redo_log	flush_after_write	true	Controls if a flush is performed after each write operation.

redo_log	automatic_recovery	true	Controls if consistencies issues of the redo log are being resolved automatically on start-up.
redo_log	write_encrypted_values	true	Controls if the data written for values is being encrypted.
redo_log	min_available_disk_space_threshold_mb	50	The minimum available disk space threshold (in MB). If the available disk space drops below the threshold the oldest log will be deleted when the log is cycled. This limitation does not apply if set to 0.
ssl	self_signed_tls_certificate_allowed	true	Specifies if self signed certificates are allowed or not.
ssl	certs_path	ca	Specifies the location the certificate verification will look for certificates.
ssl	ca_file	ca/ca-bundle.crt	Specifies the location of the CA certificates files.
ssl	cipher_list	HIGH:!AES256-SHA:!DHE-RSA-AES256-SHA:!ECDHE-RSA-AES256-SHA:!CAMELLIA:!aNULL	Specifies the cipher list string. See https://www.openssl.org/docs/man1.0.2/apps/ciphers.html for more information.
ssl	verify_peer	true	Specifies if the certificate of the peer should be verified.
logging	log_level	info	The log level to use. Can be one of fatal,error,warning,info,debug.
logging	debug_level	no	Sets the debug log level. Can be one of NO,L1,L2,L3,L4,L5.
	workers	Number of cores	Sets the number of worker threads to use for processing messages.
https	enabled	true	Specifies if https is enabled.
https	port	8463	Specifies the port to use.
https	protocol	dualstack	Specifies the protocol to use. Dualstack support ipv4 and ipv6. One of dualstack,ipv6,ipv4.
https	certName	server.pem	Specfies the name of the server certificate file. In contrast to the C Broker, the C++ Broker supports to specify the certificate with a complete path. Nevertheless, the C++ Broker will check the certs directory used by the C Broker for cerificates, if no path was specified.
https	certKeyName	key.pem	Specifies the name of the server key file. In contrast to the C Broker, the C++ Broker supports to specify the key with a complete path. Nevertheless, the C++ Broker will check the

			certs directory used by the C Broker for keys, if no path was specified.
https	cert_chain_file	server.ca-bundle	The ca file to use to validate certificates. Can be specified with a complete path.
https	tmp_dh_file	dhparams.pem	The dhparams file to use. Can be specified with a complete path.
https	cipher_list	HIGH:!AES256-SHA:!DHE-RSA-AES256-SHA:!ECDHE-RSA-AES256-SHA:!CAMELLIA:!aNULL	Specifies the cipher list string. See https://www.openssl.org/docs/man1.0.2/apps/ciphers.html for more information.
http	enabled	false	Specifies if https is enabled.
http	port	8100	Specifies the port to use.
http	protocol	dualstack	Specifies the protocol to use. Dualstack support ipv4 and ipv6. One of dualstack,ipv6,ipv4.

	defaultPermission	null	Specifies the default permissions for the connected links. Recommended setting:
			[
			`[":config","config"],`
			`[":write","write"],`
			`[":read","read"],`
			`[":user","read"],`
			`[":trustedLink","config"],`
			`["default","none"]`
			]

manager.json example and parameters.

```
{
    "enabled_links": {
        "modbus": true,
        "System": true,
        "Serial": true
    }
}
```

Option	Default	Description

enabled_links		Will be managed automatically by the lifecycle manager.
		Lists the installed links. Each link can be enabled or disabled. Example:
		`"enabled_links": {`
		`  "dataflow": true,`
		`  "c-serial": false`
		`}`

link_repository_url	https://dsa.s3.amazonaws.com/links/links.json	Url from which to download the link repository. Has to be a https address schema.
log_dir_path	logs	Where to put the link and broker log files
ssl_verify_peer	true	If the link_repository_url ssl certificates shall be verified.
certs_path	/etc/ssl/certs	Specifies the location the certificate verification will look for certificates.
ca_file	/etc/ssl/certs/ca-bundle.crt	Specifies the location of the CA certificates files.

configs		Will be managed automatically by the lifecycle manager.
		Individual link configs overridden by user settings. Example:
		`"configs": {`
		`  "Responder": {`
		`    "log": "info"`
		`  }`
		`}`

broker	depends on the brokers http and https configuration	Will be managed automatically by the lifecycle manager. The broker url to which the links shall connect.

upstream.json example and parameters.

```
{"name": "efmFogNode", "brokerName": "efmIR829edge", "url": "https://192.168.14.101:443/conn",
"enabled": true}
```

The upstreams are normally managed by the C++ Broker, but if need a file can be put into the `upstream` folder in the broker folder.

Option	Description

brokerName	The name of the current broker that will be shown under the downstream node of the upstream broker.
enabled	If this upstream is enabled or not. One of true, or false.
group	The permission group that the current broker will give to the upstream broker.
name	The name of the upstream broker, must be same as the file name in the upstream folder. This name will be shown under the upstream folder of this broker. It will also be shown under the `/sys/upstream` node.
token	The token to be used by the connecting broker when it connects to upstream broker.
url	The connection url of the upstream broker.

Obtaining documentation and submitting a service request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.