IOx Components Configuration Guide

Kinetic - Edge & Fog Processing Module (EFM) 1.7.0

Revised: July 18, 2019

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).

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 package provide the following packages:

For the Cisco IR8x9, IC3K and CGR1000 with compute module platform:

- efm_1.7.0_ic3k.tar.gz for IC 3000
- efm_1.7.0_cgr1kcm.tar.gz for CGR1000 with compute module
- efm_1.7.0_ir8x9.tar.gz for IR8x9
- The LXC Package suitable for installation contains the following components 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 1.7 or 1.8, for CGR1K and IC3K it is IOx version 1.7.

Please note that the IOx Application Health Monitoring feature requires a minimum IOx version 1.8.

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**.

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

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:
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
```

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:

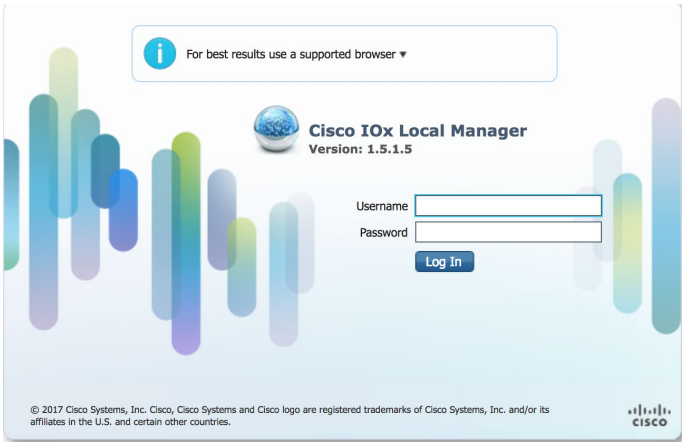
```
ip dhcp pool iox-efm-eth0-static
 host 192.168.4.2 255.255.255.0
 client-identifier 6566.6d
 default-router 192.168.4.1

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.

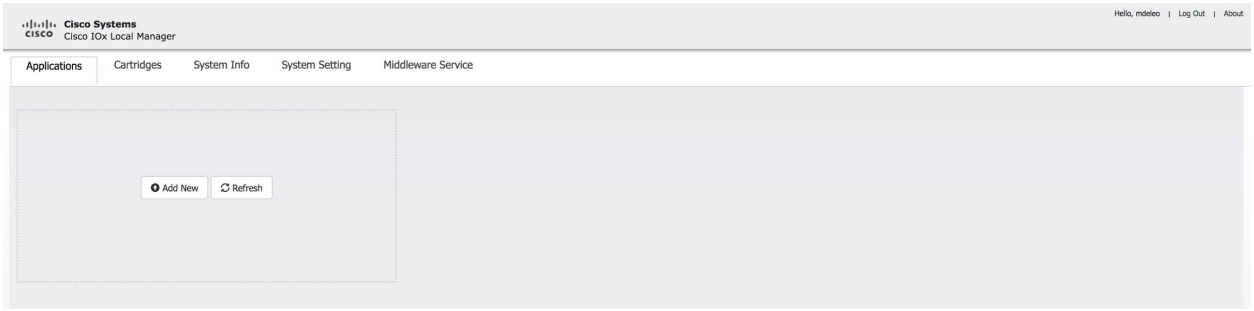
Deploying with Cisco IOx Local Manager

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

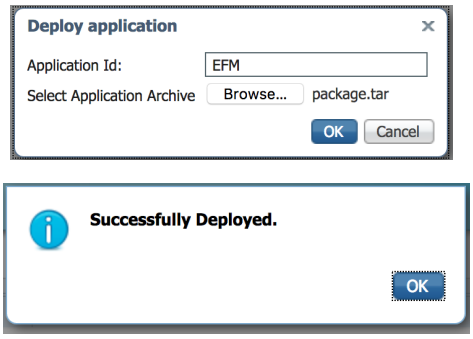
1. Connect via a web browser to the IOx router to the defined port for the Cisco IOx Local Manager. For example: https://192.168.25.201:8443/. Log in with the router credentials.



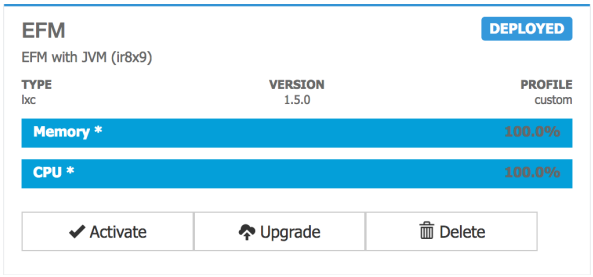
2. Click **Add New** to install the EFM application package.



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.

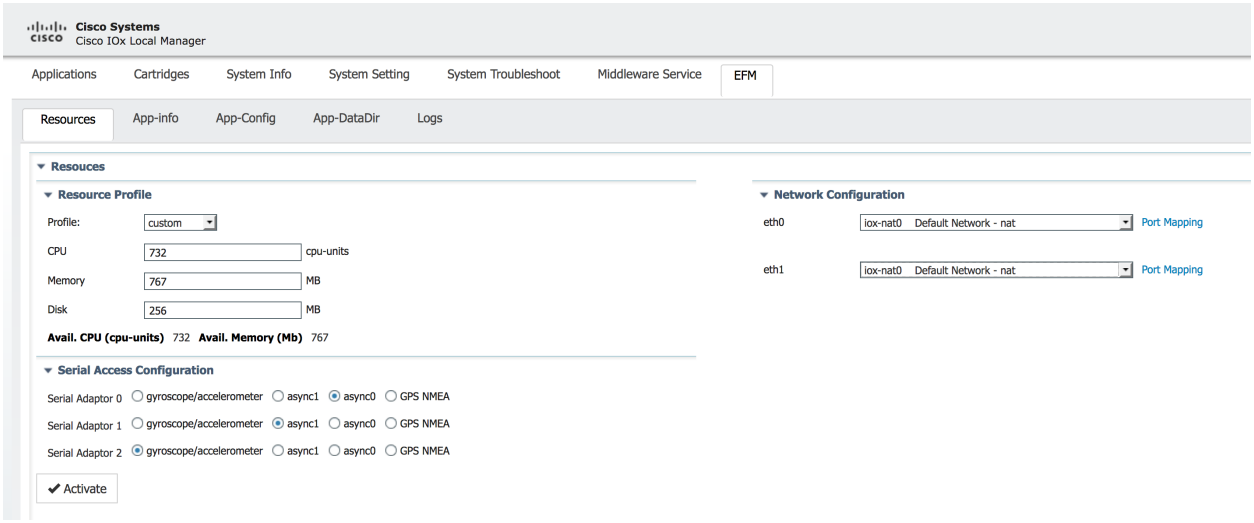


The following application status should display:



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.

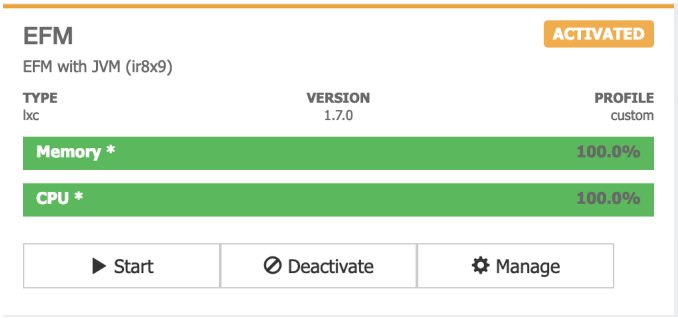


5. Serial ports need to be defined to proceed with the activation, even if the application will not communicate with any serial devices. There are no default selections.

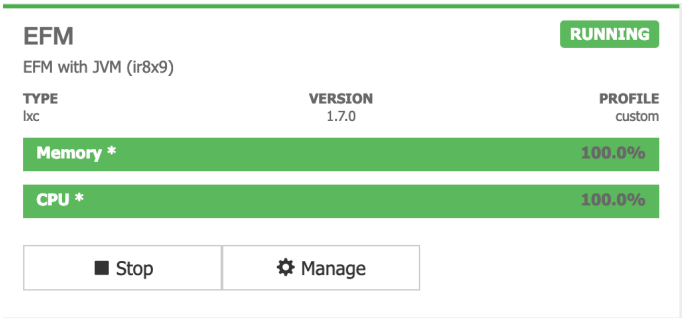
6. Ensure that the Network Configuration is set to “iox-nat0 Default Network – nat,” if using NAT. Port Mapping should be left as is for auto.

7. Click **Activate**.

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



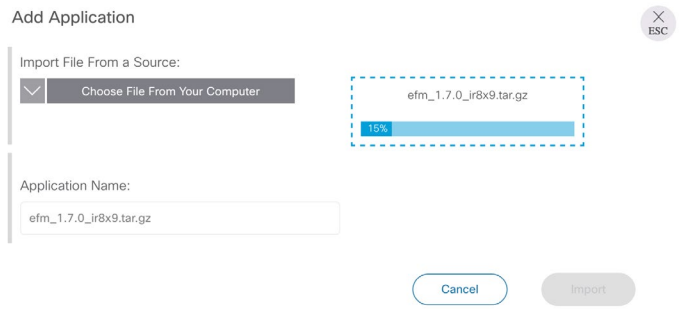
9. Now that application is activated, it must be started. Click **Start**.



Deploying with Cisco Kinetic Gateway Management Module (GMM)

Deploying EFM using GMM:

1. All applications to be deployed via GMM first must be uploaded first. Afterwards, they can be deployed to one or more gateways. Under “Applications”, click on “+ Add application” and select the EFM file to upload. Ensure that the EFM package is for ir8x9, for example efm_1.7.0_ir8x9.tar.gz. It may take a while for the application to be available for installation.



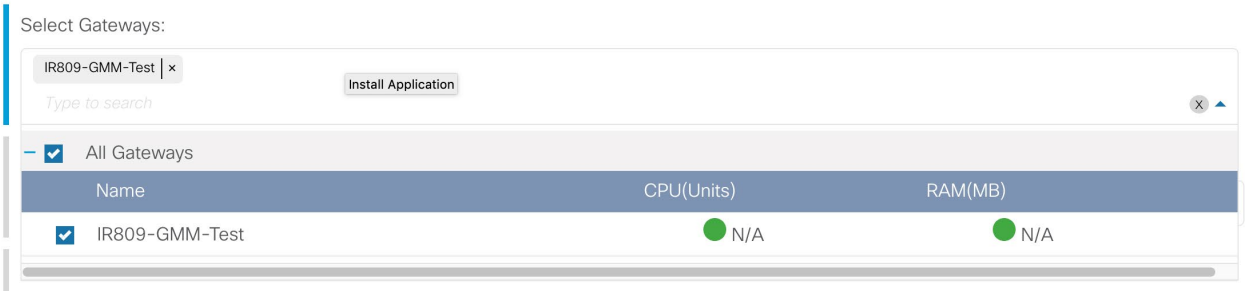
2. Once the application has been loaded into GMM, select “efm_ir8x9” in the application list. Click on “Install”. The installation configuration page will appear. By default, the following is defined:
   - Custom profile with 732 MB CPU (Max) and 767 MB RAM (Max)
   - Interface Name: Eth1 with IOx-Bridge for local networking connectivity. The EFM Broker is exposed for connectivity on the IP address with port 8080 and 8484.
   - Serial interface /dev/ttyS3
3. If the user desires to define bootstrap configuration, then select “Application Specific Parameters”. The following can be configured,

   The application specific parameters definition allows the EFM application package to evaluate and apply (if needed) configuration parameters at the start or restart of the EFM IOx application. Three optional configuration sections are supported:

   - Broker_upstream: The first section will define a single upstream connection to be used for the broker, outside of the existing upstream definition files in the upstream folder. The output is stored in the file upstream/package_config_upstream.

- Broker_ssl: The second section will allow to set the brokers SSL certificate and key to be used for the SSL connections.
- Sys_config: The third section allows for the definition of one or more user defined tags that are placed to the broker node structure under the /downstream/sys/cisco/ini. The broker expects a JSON table or a string, if no JSON table is introduced it defaults to a string value.

4. Select Gateways to install by choosing one or more gateways under “Select Gateways”. Note that only the available gateways in good health appear. Ensure that the



5. Select “Install” to deploy. After a period of time, GMM will deploy the application for use.

EFM is automatically activated and started using GMM.

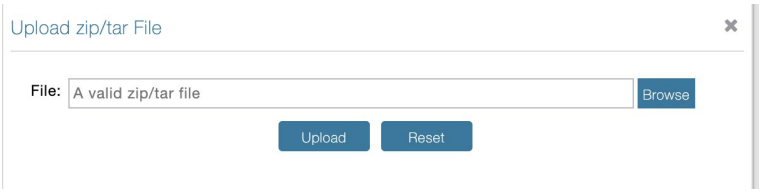
Once installed, it is possible to review the Application Detail on Instance of each gateway by selecting the gateway, select the apps tab and click on the gateway in the list. The IP address and interface name are listed for incoming broker connections on the port list (default 8080 and 8484 for http/https connections respectively).

Deploying with Cisco IoT Field Network Director (IoT-FND)

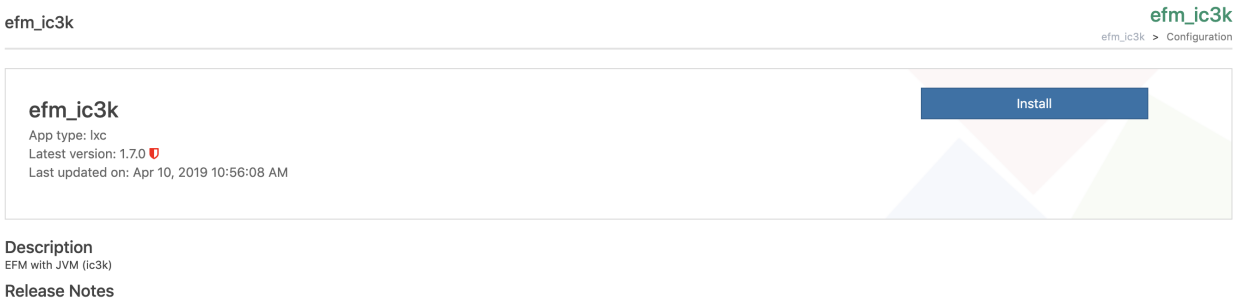
Deploying EFM using IoT-FND:

1. All applications to be deployed via IoT-FND first must be uploaded first. Afterwards, they can be deployed to one or more gateways. Under “Applications”, click on “+ Add application” and select

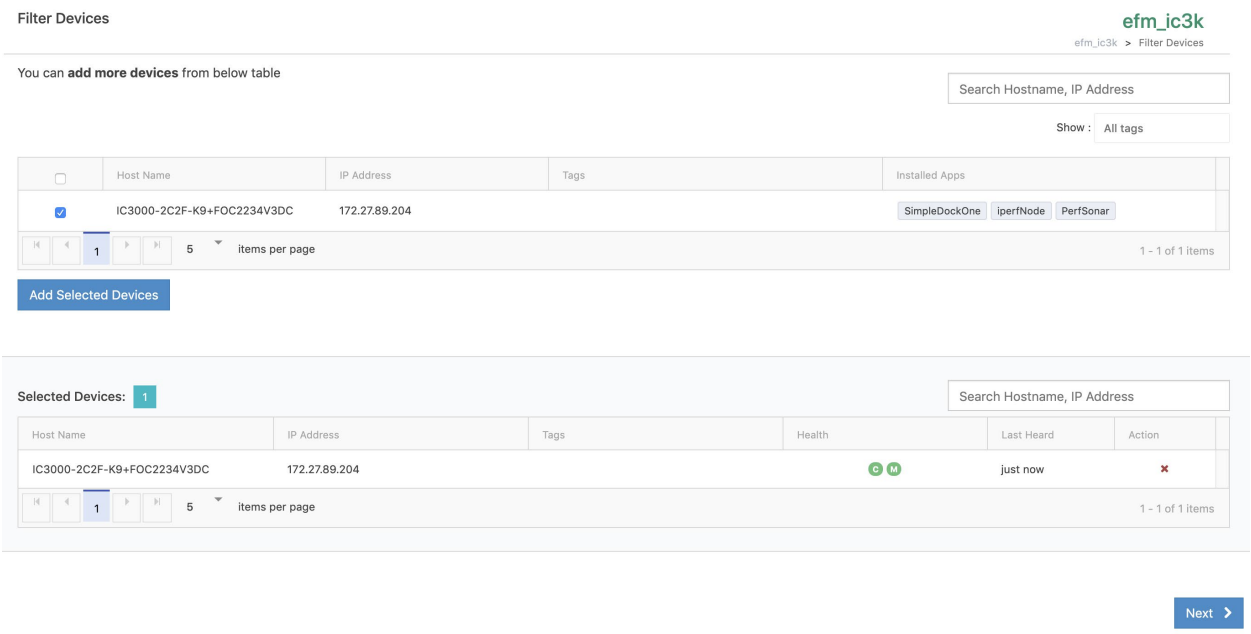
the EFM file to upload. Ensure that the EFM package is for ir8x9, for example efm_1.7.0_ic3k.tar.gz. It may take a while for the application to be available for installation.



2. Once the application has been loaded into IoT-FND, select “efm_ic3k” in the application list. Click on “Install”. Select the devices to install the EFM software. Select Add Selected Devices. Click on “Next” to continue.

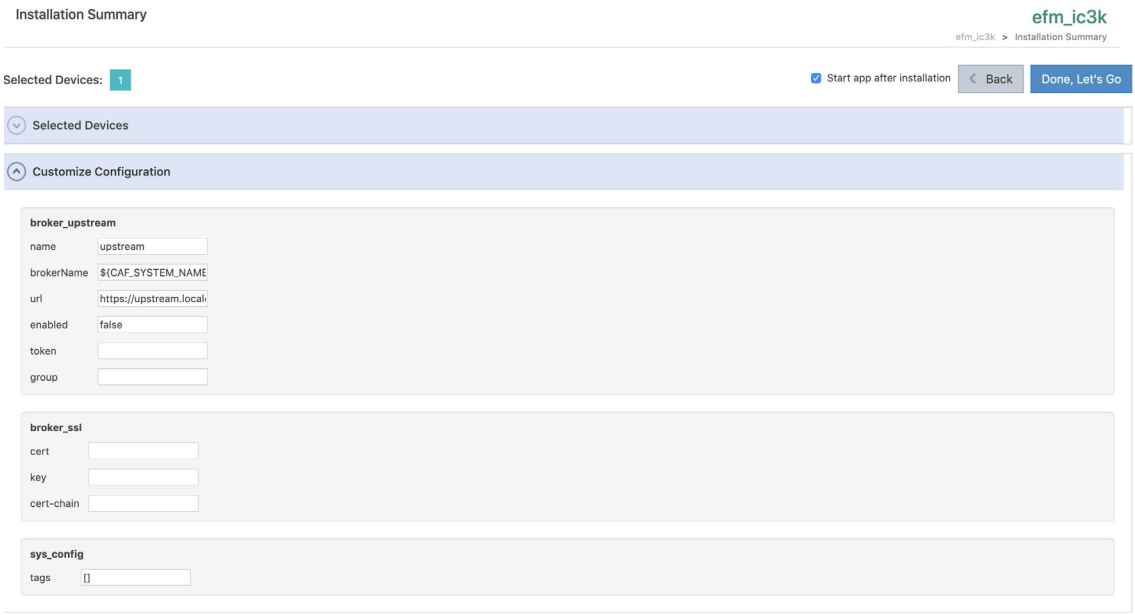


3. Select the gateway devices to install EFM upon. Then select Next to continue.



4. The Installation Summary page will appear. This allows for confirmation of the devices, customize the EFM application configuration and other optional features.

5. If the user desires to define bootstrap configuration, then expand the “Customize Configuration” bar. The following can be configured,

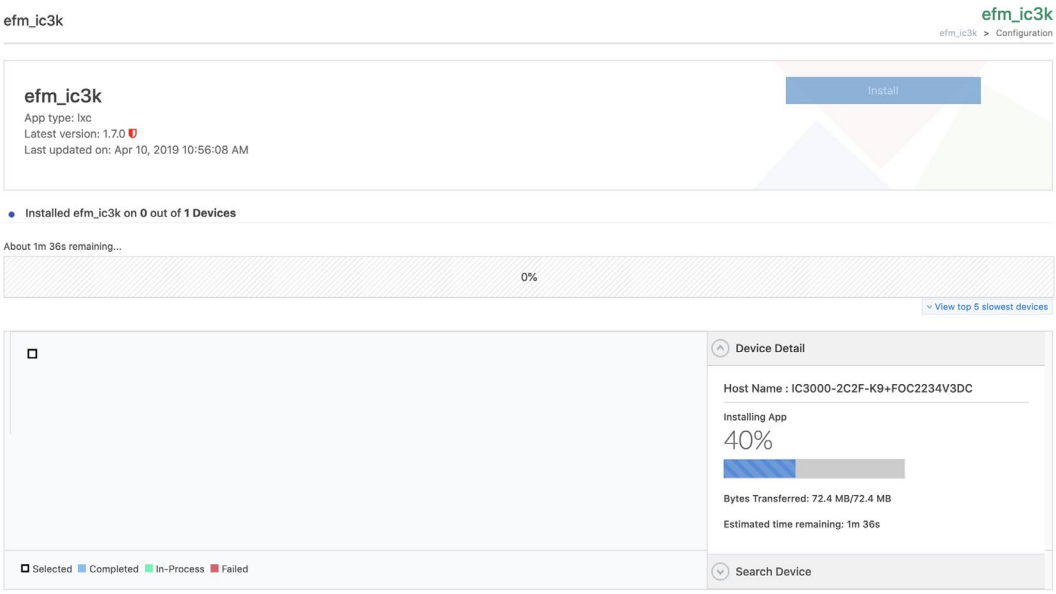


The application specific parameters definition allows the EFM application package to evaluate and apply (if needed) configuration parameters at the start or restart of the EFM IOx application. Three optional configuration sections are supported:

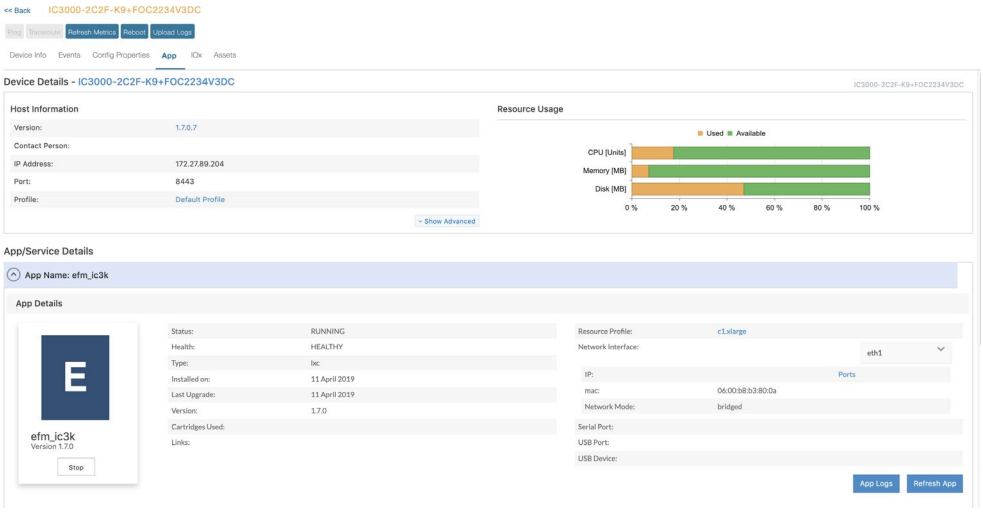
- Broker_upstream: The first section will define a single upstream connection to be used for the broker, outside of the existing upstream definition files in the upstream folder. The output is stored in the file upstream/package_config_upstream.
- Broker_ssl: The second section will allow to set the brokers SSL certificate and key to be used for the SSL connections.
- Sys_config: The third section allows for the definition of one or more user defined tags that are placed to the broker node structure under the /downstream/sys/cisco/ini. The broker expects a JSON table or a string, if no JSON table is introduced it defaults to a string value.

6. Ensure that the Resource Profiles and VCPUs are configured and no red alerts are still shown prior to deployment.

7. Select “Done, Let’s Go” to deploy. After a period of time, IoT-FND will deploy the application for use.



IoT-FND will start the application for use. Under Devices, select the Field Devices, select the device with the EFM image installed.



Once installed, it is possible to review the Host Information for the gateway in the Device Details of the page above The IP address and interface name are listed for incoming broker connections on the port list (default 8080 and 8484 for http/https connections respectively).

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
219.129
```

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.

The usage of QoS level 3 (persistent queue) subscriptions to brokers or links running on the IR8x9 series devices is not supported due to the high write volume data to the flash memory, that will substantially reduce the devices lifetime. If such devices have been deployed in your environment, you should check every QoS level 3 subscriptions target to ensure not to subscribe to above mentioned devices.

It is currently not possible the ability to deactivate QoS level 3 subscription in brokers or links.

Nevertheless, it is possible to attach external non-flash storage to at least some of the above-mentioned devices. To use QoS level 3 subscriptions, you have to configure the external storage as persistent queue storage. The path to the mass storage for persistent queues can be configured only for C/C++ Broker and C++ Links. See the 'Configuring the EFM C++ Message Broker configuration files' section in this document how to configure your devices.

The QoS level 3 restriction does not apply to devices of the IC3000 series or the CGR1000 Compute Module.

Environmental variables and user tags read at startup

With the new system dslink, environmental variables are read from the IOx container as well as the optional user defined tags and are placed under the broker node tree /downstream/sys/cisco.

The IOx container environmental variables are read at startup and placed under the broker node tree the /downstream/sys/cisco/environment. See the IOx documentation for a detailed description at https://developer.cisco.com/docs/iox/#!application-development-concepts/application-development-concepts under the Environmental Variables section.

User tags, if any, that are defined in the bootstrap process appear under the broker node tree /downstream/sys/cisco/ini.

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
- 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.

efmFogNode.json example and parameters.

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

The upstream brokers managed by the C++ Broker, but if need a file can be put into the `upstream` 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.

Configuring the EFM C++ Message Broker Upstream

To simplify the EFM C++ Message Broker deployment, in many cases it is desirable that an upstream message broker be defined before the start of the application.

There are two manners of defining the upstream broker without using the administrative tools. The first is placing an upstream definition file into the upstream folder. The other alternative is to use the bootstrapping mechanism, either through the Local Manager, GMM, FND or ioxclient for the applicable supported platforms.

Defining and uploading an upstream broker definition file

Generating the upstream connection definition file

For every broker to broker connection in the EFM we need to define the following attributes:

- The Upstream broker name on this connection[2]
- The Local Broker name on this connection
- The URL to be used for the upstream connection. All broker to broker URL must end in "/conn". For example, https://192.168.14.101:443/conn. [3]
- Define if the connection is enabled at startup. True for enabled, false for disabled.

A text file needs to be created with these attributes in the following format json structure. The name of the file is the same as the Upstream broker name for consistency. Replace the following placeholders:

- <upstreamName> with Upstream broker name on this connection
- <localName> with Local Broker name on this connection
- <upstreamURL> with the upstream broker URL, including the port and terminating in "/conn".

```
{"name": "<upstreamName>", "brokerName": "<localName>", "url": "<upstreamURL>", "enabled": true}
```

[2] In the EFM and DSA IoT architecture, the broker names defined in a connection are not unique to the node, but only used per connection. They must be unique names between the same brokers.

[3] The URL can use a numeric IP address. The current EFM IOx package does not contain an /etc/resolv.conf or obtain it from the GuestOS and therefore cannot use DNS for name resolution.

For example, file name: efmFogNode

Each upstream message broker is defined by a unique file under the App-DataDir `upstream` folder. It is suggested that the name of the file correspond to the broker name. The file can be uploaded using the LocalManager or ioxclient after activation.

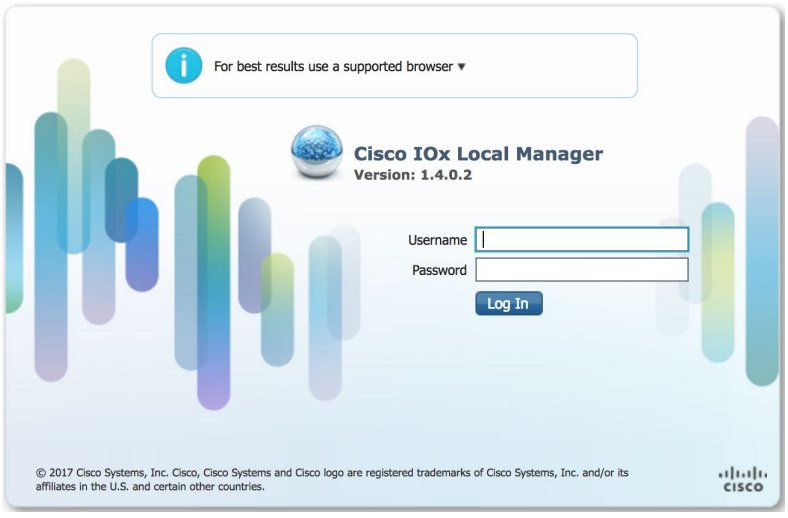
In this example file efmFogNode, an upstream broker efmFogNode is defined as the URL https://192.168.14.101:443 from the local broker name efmIR829edge.

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

There are two ways to modify the EFM application configuration, the first using the Local Manager UI and the second is the ioxclient. The ioxclient allows for automation through scripting if needed.

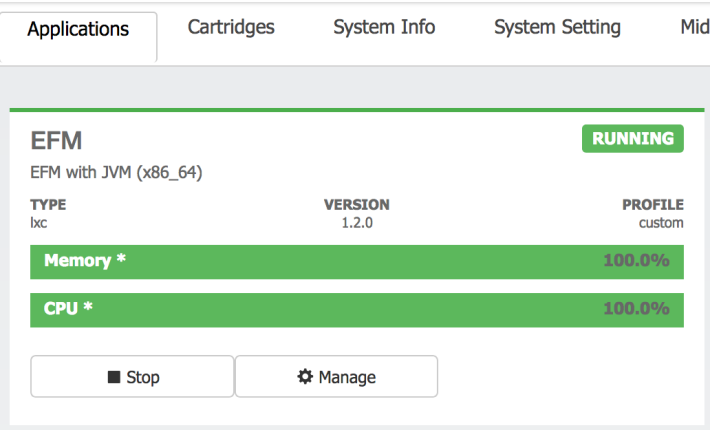
Installing and upstream broker connection file using the Local Manager

Connect via a web browser to the IOx router to the defined port for the Cisco IOx Local Manager.



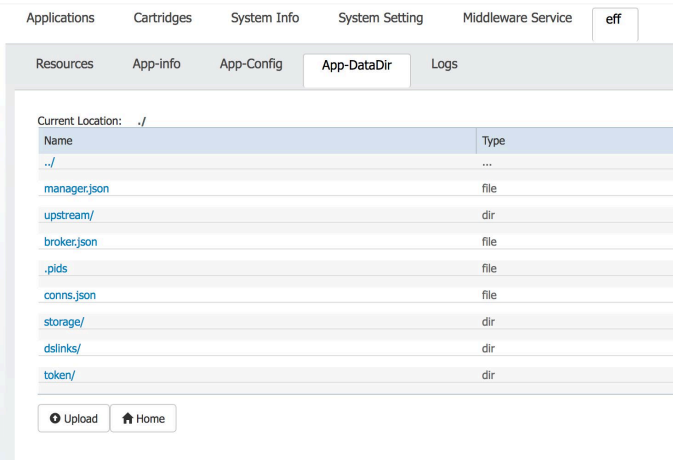
For usage of the Local Manager UI you have to use a browser to connect to the IOx device using the https connection to the port 8443. You will have to log into the UI using the required credentials.

Once logged in, the interface should show the Applications by selecting the Applications tab.



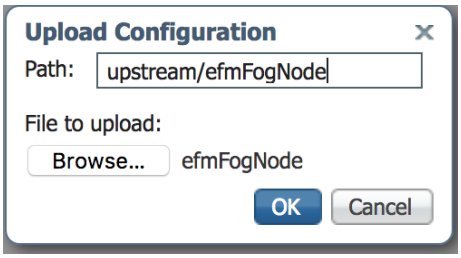
The EFM application is assumed to be installed and running, but before we can make changes to the broker configuration file, we must first stop the application. Simply select **Stop** in the Local Manager UI.

After the application has stopped, select the application name “EFM” in the tabs. This will expose additional tabs underneath to allow for more functions. Select the “App-DataDir” tab.



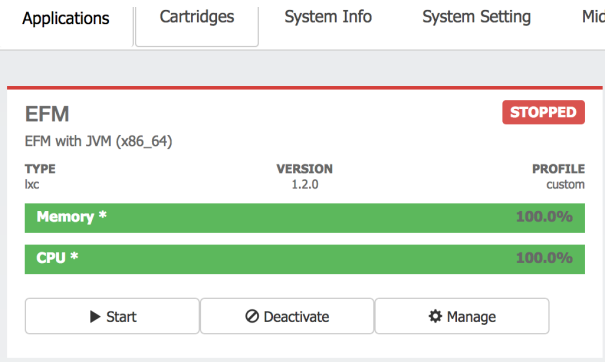
This application directory App-DataDir contains all configuration files and installed links.

First upload the server certificate by selection of the **Upload** button below the file list. Enter the target path **upload/efmFogNode** into the Path edit field of the pop-up window. Pay attention not to forget the folder name in the path. Then hit the **browse** button in the pop-up window and select the server certificate file in your local file system. After selecting of the **Ok** button, the Local Manager will create a folder named **efmFogNode** in the upload directory.



To verify the files have been uploaded, you may check the content of the upload folder by selecting the **upstream** folder in the UI.

Now we start the application by select the **Applications** main management tab and the **Start** button for the EFM application.



The broker will now start and try to connect the upstream broker as defined in the upstream file efmFogNode.

To verify that the broker is connected or connecting to the upstream broker, you may check the broker log file **broker.log** from the Logs tab inside the **EFM** management tab. There should line indicating that it was successfully connected to the upstream broker. An alternative is using the EFM System Administrator, selecting the upstream broker in the Brokers pane and expanding to see if the edge broker is shown underneath.

```
2017-12-19 21:07:11 INFO [upstream] - Successfully connected to the upstream broker 'efmFogNode'
```

Installing and upstream broker connection file using the ioxclient

The EFM application is assumed to be installed and running, before we can make changes to the broker configuration file, we must first stop the application. To stop the application type:

```
ioxclient application stop EFM
```

Upload the upstream broker connection file using the ioxclient command:

```
ioxclient application appdata upload EFM efmFogNode upstream/efmFogNode
```

Now we start the application:

```
ioxclient application start EFM
```

To verify that the broker is connected or connecting to the upstream broker, you may check using the Local Manager the broker log file **broker.log** from the Logs tab inside the **EFM** management tab. There

should line indicating that it was successfully connected to the upstream broker. An alternative is using the EFM System Administrator, selecting the upstream broker in the Brokers pane and expanding to see if the edge broker is shown underneath.

```
2017-12-19 21:07:11 INFO [upstream] - Successfully connected to the upstream broker 'efmFogNode'
```

The upstream brokers managed by the C++ Broker, but if need a file can be put into the `upstream` folder.

Defining an upstream broker, ssl certificate or user defined tag using the bootstrap process

The bootstrap process allows the EFM application package to evaluate and apply (if needed) configuration parameters at the start or restart of the EFM IOx application. Three configuration sections are supported:

1. The first section will define a single upstream connection to be used for the broker, outside of the existing upstream definition files in the upstream folder. The output is stored in the file upstream/package_config_upstream.
2. The second section will allow to set the brokers SSL certificate and key to be used for the SSL connections.
3. The third section allows for the definition of one or more user defined tags that are placed to the broker node structure under the /downstream/sys/cisco/ini. The broker expects a JSON table or a string, if no JSON table is introduced it defaults to a string value.

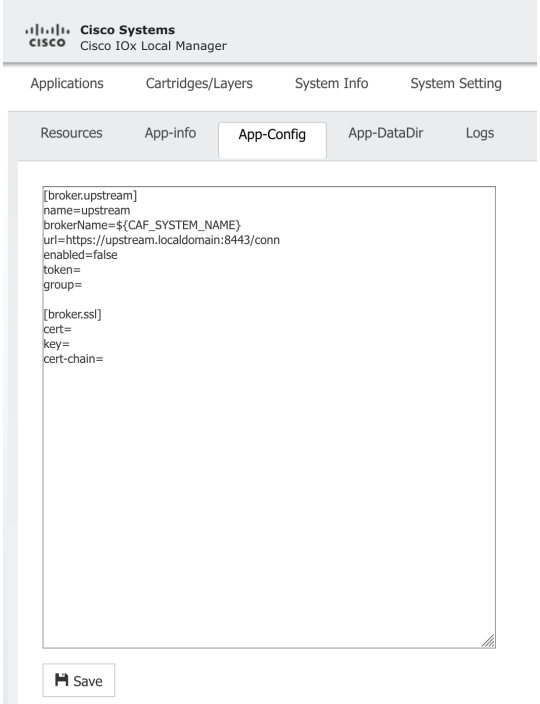
Defining the upstream broker, ssl certificate or user defined tag using app-Config through the Local Manager

Before starting or restarting the EFM application, login to the Local Manager select the App-Config tab for the EFM application that was installed. The blank template will appear as follows:

```
[broker.upstream]
name=upstream
brokerName=${CAF_SYSTEM_NAME}
url=https://upstream.localdomain:8443/conn
enabled=false
token=
group=

[broker.ssl]
cert=
key=
cert-chain=
```

If the url parameter is empty or missing, the upstream definition file will not be touched. If one of the following parameters is missing, the following default values will be used:

```
name=upstream
brokerName=unspecified-iox-device
enabled=true
```



For example, to define the upstream name efmFogNode, the local brokerName efmIR829 with the URL https://192.168.14.101:443 the following input will work. Note that the attribute enabled must be true for the settings to operate.

Click **Save** and confirm. Start the EFM application.

Defining the upstream and/or ssl certificate using the ioxclient

Before starting or restarting the EFM application, the bootstrap start file configures the EFM upstream and/or certificate. The ioxclient setconfig option allows to upload the bootstrap startup file.

The template text file is the following.

```
[broker.upstream]
name=upstream
brokerName=${CAF_SYSTEM_NAME}
url=https://upstream.localdomain:8443/conn
enabled=false
token=
group=

[broker.ssl]
cert=
key=
cert-chain=
```

If the url parameter is empty or missing, the upstream definition file will not be touched. If one of the following parameters is missing, the following default values will be used:

```
name=upstream
brokerName=unspecified-iox-device
enabled=true
```

To upload the boostrap file, use the following command:

```
ioxclient app setconfig <app-name> <bootstrap-file-name>
```

For example, if the application is named EFM and the text file is EFM-package_config.ini the command would be the following:

```
ioxclient app setconfig EFM EFM-package_config.ini
```

To verify the boostrap configuration, use the following command. The application in this example is named EFM.

```
ioxclient app getconfig EFM
```

Configuring the SSL certificate to allow for secure inbound connections on EFM IOx app

A necessary step for the message broker to allow incoming secure connections is to properly install the SSL certificate files and modify the broker configuration file server.json. The IOx application is configured with a self-signed certificate and should work without modifications. But if the administrator desires to change the configuration, this process describes the steps.

The process described below only works on versions 1.1 and greater due to internal file structure are different and the configuration files are not located in the data partition of the application container.

Generating a Self-Signed Key and Certificate

For some environments, the use of a self-signed certificate is sufficient. For production, the user can obtain certificates and install them on the EFM system.

In the following installation description, we refer to the server key file as *key.pem* and the server certificate file to be *server.pem*

On any posix compliant system with an openssl installation, the following command will generate a key file and a corresponding self-signed certificate:

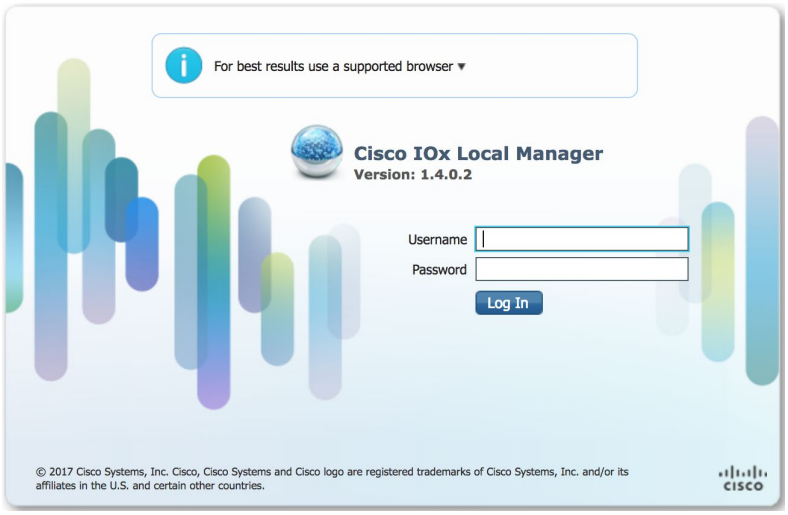
```
openssl req -nodes -x509 -days 365 -newkey rsa:2048 -out server.pem -keyout key.pem -subj
"/C=US/ST=CA/L=San Francisco"
```

Configuring the EFM application to support inbound of HTTPS connections

There are two ways to modify the EFM application configuration, the first using the Local Manager UI and the second is the ioxclient. The ioxclient allows for automation through scripting if needed.

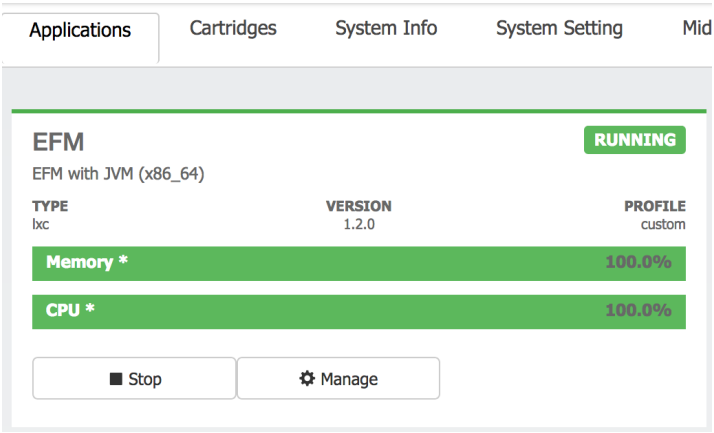
Installing and Configuring Self-Signed Key and Certificate using the Local manager

Connect via a web browser to the IOx router to the defined port for the Cisco IOx Local Manager.



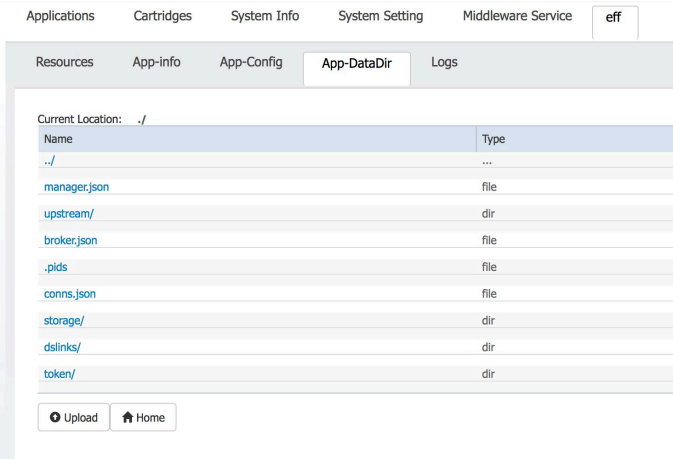
For usage of the local manager UI you have to use a browser to connect to the IOx device using the https connection to the port 8443. You have to log into the UI using the required credentials and the

Once logged in, the interface should show the Applications by selecting the Applications tab.



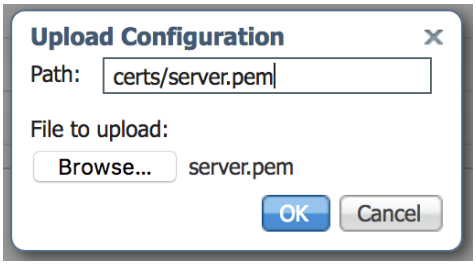
The EFM application is assumed to be installed and running, before we can make changes to the broker configuration file, we must first stop the application. Simply select **Stop** in the Local Manager UI.

After the application has stopped, select the application name “EFM” in the tabs. This will expose additional tabs underneath to allow for more functions. Select the “App-DataDir” tab.

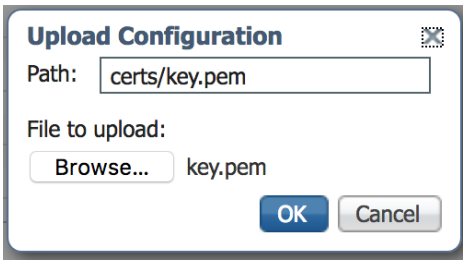


This application directory App-DataDir contains all configuration files and installed links.

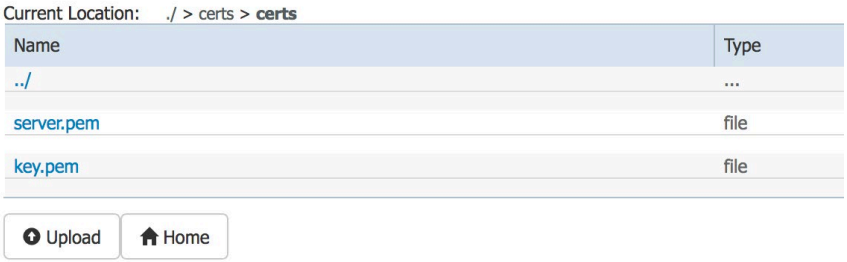
First upload the server certificate by selection of the **Upload** button below the file list. Enter the target path **certs/server.pem** into the Path edit field of the pop up window. Pay attention not to forget the folder name in the path. Then hit the **browse** button in the pop up window and select the server certificate file in your local file system. After selecting of the **Ok** button the Local Manager will create a folder named **certs** in the application directory and should have put the server certificate **server.pem** into that folder.



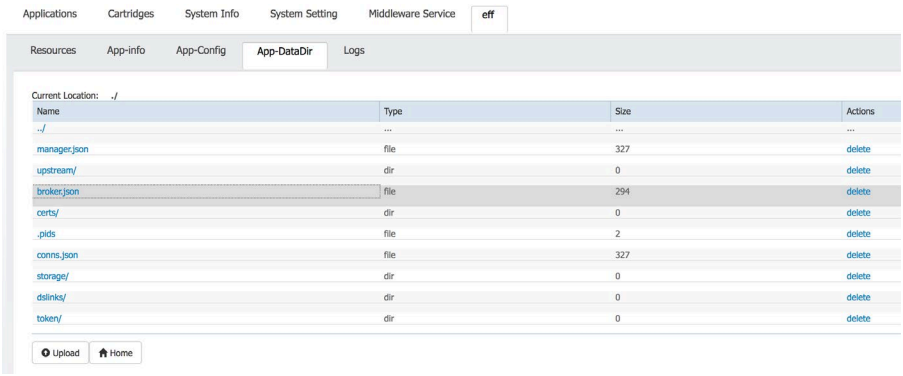
In the next step, we upload the server key file the same way as we uploaded the certificate. Select the **upload** button a second time and enter **certs/key.pem** into the target path edit field this time. Then select the **Browse** button and select the server key file in your local file system. After selection of the **Ok** button the local manager should have added the key file into the **certs**, too.



To verify the files have been uploaded, you may check the content of the certs folder by selecting the **certs** folder in the UI. But don't forget to get back to the application directory for the next steps. The certs folder should look like in the next screen shot.



We need to modify the existing broker.json configuration file by downloading, editing and replacing the existing file. To download the configuration file **broker.json,** select it in the file list and store it locally our your computer.



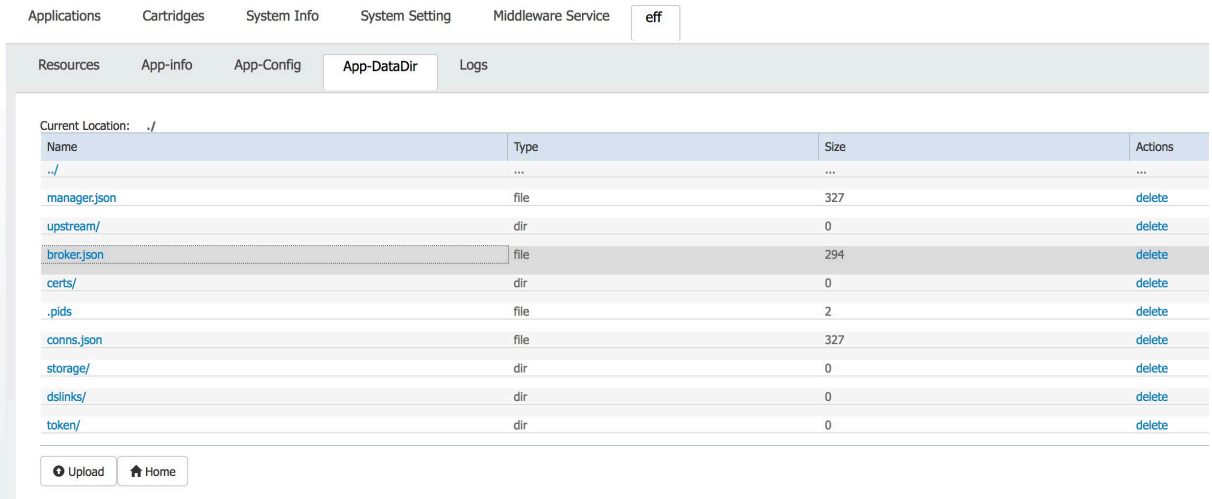
Now modify it with an editor of your choice. Take care the file should still be a plain ASCII file with a valid json contents after editing. The following green marked elements needed to be modified in the configuration file.

- In the https section, the value of **enabled** must be set to **true.**
- In the https section, add the name of the certificate file **“certName”: “server.pem”**
- In the https section, add the name of the key file **“certKeyName”: “key.pem”**

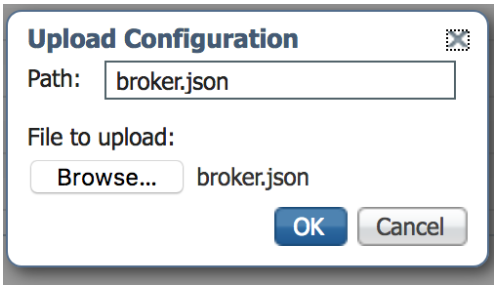
After the modification, the configuration file should look like the following (pay attention to maintain the JSON formatting!):

```
{
  "http": {
    "enabled": true,
    "host": "0.0.0.0",
    "port": 8080
  },
  "https": {
    "enabled": true,
    "host": "0.0.0.0",
    "port": 8484,
  "certName": "server.pem",

  "certKeyName": "key.pem"

  },
  "log_level": "info",
  "allowAllLinks": true,
  "maxQueue": 1024,
  "defaultPermission": null,
  "storage": {
    "path": "."
  }
}
```

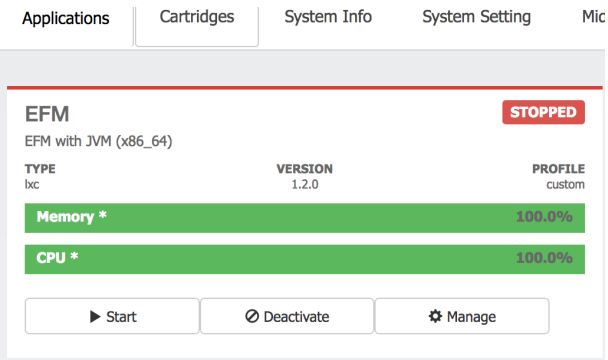
To upload the modified configuration file, we first need to delete the original file since the Local Manager does not allow to overwrite an existing file. Therefore we must first select the **delete** option on the rightside of the broker.json config file to delete the file on the device.



Now you may upload the modified configuration file from your local file system by selecting the **Upload** button below the file list one more time. Enter **broker.json** = into the Path: field. Then select your modified configuration file by selecting the **Browse** button. Hit **Ok** and the modified configuration file should show up in the file list.



Now we start the application by select the **Applications** main management tab and the **Start** button for the EFM application.



The broker will now start and in addition to TCP port 8080, will now listen port 8484. Port TCP 8080 and 8484 correspond to http and https connections respectively. To verify that the broker is listening, you may check that the configuration is work correctly by downloading the broker log file **broker.log** from the Logs tab inside the **EFM** management tab. There should now be a second log info line starting the HTTPS server additionally to the HTTP server.

```
2017-11-08 10:08:45 INFO [server] - HTTP server bound to 0.0.0.0:8080
2017-11-08 10:08:45 INFO [server] - HTTPS server bound to 0.0.0.0:8484
```

Installing and Configuring Self-Signed Key and Certificate using the ioxclient

The EFM application is assumed to be installed and running, before we can make changes to the broker configuration file, we must first stop the application. To stop the application type:

```
ioxclient application stop EFM
```

In the next step, we upload the server key file the same way as we uploaded the certificate. Select the **upload** button a second time and enter **certs/key.pem** into the target path edit field this time.

Start by uploading the server certificate file using the ioxclient command:

```
ioxclient application appdata upload EFM server.pem certs/server.pem
```

Next, we upload the server key file using:

```
ioxclient application appdata upload EFM key.pem certs/key.pem
```

We need to modify the existing broker.json configuration file by downloading, editing and replacing the existing file. To download the configuration file **broker.json,** select it in the file list and store it locally our your computer.

```
ioxclient application appdata view EFM broker.json
```

Now modify it with an editor of your choice. Take care the file should still be a plain ASCII file with a valid json contents after editing. The following green marked elements needed to be modified in the configuration file.

- In the https section, the value of **enabled** must be set to **true.**
- In the https section, add the name of the certificate file **“certName”: “server.pem”**
- In the https section, add the name of the key file **“certKeyName”: “key.pem”**

After the modification, the configuration file should look like the following (pay attention to maintain the JSON formatting!):

```
{
  "http": {
    "enabled": true,
    "host": "0.0.0.0",
    "port": 8080
  },
  "https": {
    "enabled": true,
    "host": "0.0.0.0",
    "port": 8484,
  "certName": "server.pem",

  "certKeyName": "key.pem"

  },
  "log_level": "info",
  "allowAllLinks": true,
  "maxQueue": 1024,
  "defaultPermission": null,
  "storage": {
    "path": "."
  }
}
```

To upload the modified configuration file, we first need to delete the original file since the ioxclient does not allow to overwrite an existing file.

```
ioxclient application appdata delete EFM broker.json
```

Now you may upload the modified configuration file from your local file system:

```
ioxclient application appdata upload EFM broker.json broker.json
```

Now we start the application:

```
ioxclient application start EFM
```

The broker will now start and in addition to TCP port 8080, will now listen port 8484. Port TCP 8080 and 8484 correspond to http and https connections respectively. To verify that the broker is listening, you may check that the configuration is work correctly by downloading the broker log file **broker.log** from the Logs tab inside the **EFM** management tab. There should now be a second log info line starting the HTTPS server additionally to the HTTP server.

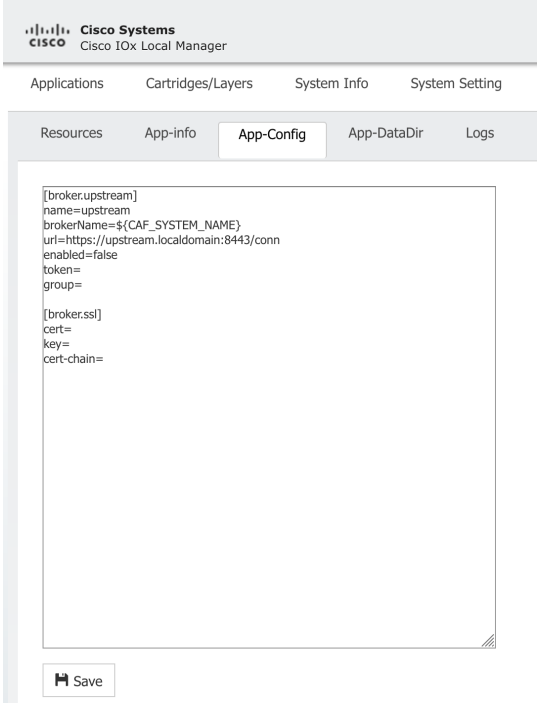
```
ioxclient application logs download broker.log

2017-11-08 10:08:45 INFO [server] - HTTP server bound to 0.0.0.0:8080
2017-11-08 10:08:45 INFO [server] - HTTPS server bound to 0.0.0.0:8484
```

Defining an SSL certificate broker using the bootstrap process

Defining the SSL certificate using app-Config through the Local Manager

Before starting or restarting the EFM application, login to the Local Manager select the App-Config tab for the EFM application that was installed. The blank template will appear as follows:

```
[broker.upstream]
name=upstream
brokerName=${CAF_SYSTEM_NAME}
url=https://upstream.localdomain:8443/conn
enabled=false
token=
group=

[broker.ssl]
cert=
key=
cert-chain=

[tags]
```



As no line breaks are possible in the config file the required certificate and key file line breaks have to be excaped using a \n.

For example, to define the ssl certificate and key

name efmFogNode, the local brokerName efmIR829 with the URL https://192.168.14.101:443 the following input will work. Note that the attribute enabled must be true for the settings to operate.

Click **Save** and confirm. Start the EFM application.

```
[broker.upstream]
name=efmFogNode
brokerName=efmIR829
url=https://192.168.14.101:443/conn
enabled=true
```

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.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.