



IOx Application Configurator Guide

Kinetic - Edge & Fog Processing Module (EFM) 1.7.0

Revised: April 4, 2018

Table of Contents

IOx Application Configurator Guide	1
Kinetic - Edge & Fog Processing Module (EFM) 1.7.0	1
Introduction.....	2
Prerequisites.....	2
Downloading the application data.....	2
Setting up the configuration template	3
Defining the template script configuration	4
Generating the target's appdata.....	5
Uploading the generated data to the targets	5
Obtaining documentation and submitting a service request	6

Introduction

The EFM IOx Application Configurator script can be used to deploy configuration data for multiple IOx target platforms. This requires a reference installation that includes the configurations of the broker, the DSManager, and the links that should also be deployed on the targets.

This document describes the following:

- Prerequisites
- Downloading the configuration data from an existing platform
- Defining template placeholder in the configuration files
- Defining a configuration file including target specific variables, upstreams, and global variables
- Generating the target appdata based on these definitions
- Uploading the appdata on the target platforms

Prerequisites

- A preconfigured reference EFM installation on a supported IOx device
- Default EFM installations on the target IOx devices
- IOXClient version 1.4.0 and above
- Configured IOXClient profiles for the source and target devices
- Python version 2.7.0 and above for the configuration generator script

Downloading the application data

The first step is activating the iox profile from which want to get the data:

```
$ ioxclient profile activate ir809-dev
Active Profile : ir809-dev
Activating Profile ir809-dev
Saving current configuration
```

To find the application name of the of the existing installation, we want to use as the reference/source application:

```
$ ioxclient application list
Currently active profile : ir809-dev
Command Name: application-list
Saving current configuration
List of installed App :
1. efm          --->  RUNNING
```

Download the data mount, including the application data:

```
$ ioxclient application datamount download efm
Currently active profile : ir809-dev
Command Name: application-datamount-download
Read bytes : 2213436
Datamount Content for efm is downloaded at efm-datamount.tar.gz
```

Extract the downloaded data mount archive:

```
$ tar xf efm-datamount.tar.gz
$ cd efm
$ ls appdata
broker.json  dslinks  manager.json
```

Setting up the configuration template

Based on this appdata, the template configuration will be defined by inserting variable-placeholders into the available configuration files. A variable-placeholder is defined such as: `{{ VARIABLE }}`. The value of the variable will be defined in a separate script configuration file, which will be the description in the next section. First we will define a example broker.json configuration file and insert port variable-placeholders:

```
$ cat appdata/broker.json
{
  "http": {
    "enabled": true,
    "host": "0.0.0.0",
    "port": {{ HTTP_PORT }}
  },
  "https": {
    "enabled": false,
    "host": "0.0.0.0",
    "port": {{ HTTPS_PORT }}
  },
  "log_level": "info",
  "allowAllLinks": true,
  "maxQueue": 1024,
  "defaultPermission": null,
  "storage": {
    "path": "."
  }
}
```

```
}  
}
```

These kinds of modifications can be done to other configuration files defined in the appdata directory.

Defining the template script configuration

A configuration for the script including upstreams, allowed file extensions, and variables could look like this for the example described before:

```
$ cat script_config.json  
{  
  "default": {  
    "vars": {  
      "HTTP_PORT": 8080,  
      "HTTPS_PORT": 8443  
    }  
  },  
  "targets": {  
    "ir809-02": {  
      "upstream": [  
        "ir809-03"  
      ]  
    },  
    "ir809-03": {  
      "vars": {  
        "HTTP_PORT": 80,  
        "HTTPS_PORT": 8081  
      }  
    }  
  },  
  "upstream_template": {  
    "name": "{{upstream_name}}",  
    "brokerName": "{{name}}",  
    "url": "http://{{upstream_name}}:{{upstream_HTTP_PORT}}/conn",  
    "enabled": true  
  },  
  "template_file_extensions": [  
    "json"  
  ]  
}
```

The first section describes the default variables that are globally valid for each target device. The defaults can be overwritten by target specific variables in the "targets" section. Additionally, each target can have a list of upstreams. Each upstream must be defined in the configuration since the script uses the target variables prefixed with upstream_. These prefixed variables can be used in the "upstream_template" definition. In addition, the two variables {{ name }} and {{ upstream_name }} that provide the target's name available for the upstream_template section. template_file_extensions is a list that can be used

to restrict the template variable replacements. If this list is empty, every file of the source appdata directory will be interpreted as a config-template.

Generating the target's appdata

The basic use of the script is:

```
efm_config_manager.py config.json source/appdata dest
```

By following the steps described above, the script call would look like this:

```
efm_config_manager.py script_config.json appdata generated
```

The first step in the script is to copy the source appdata directory into subdirectories of the destination directory generated for each target. The script applies the variables to the target files with matching file extensions and generates the upstream files in generated/<target>/upstream.

Note: Each file defined in the generated directory will overwrite an existing file on the target IOx platform. Thus, the source directory that is used to generate the target files/directories should be cleaned up before generating the targets.

Uploading the generated data to the targets

To upload the generated appdata, the second script named efm_config_deploy.py is used. It takes the following three arguments:

```
efm_config_deploy.py config source application
```

When using the example, the call looks like:

```
efm_config_deploy.py script_config.json generated efm
```

This command is starting the IOXClient in the background and uploading the generated files to the IOx platform targets.

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 the following URL:

<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.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company.

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.