Contents

Introduction .................................................................................................................................... 3
Features and Functions ................................................................................................................... 3
The Edge and Fog Fabric Components ........................................................................................... 4
Hardware Requirements ................................................................................................................. 5
EFF Components Protocols and Ports ............................................................................................. 5
Licensing installation and requests ................................................................................................. 6
Required Third Party Libraries for EFF Installation on Linux .......................................................... 6
Secure Mode Operation .................................................................................................................. 6
Basic component installation scenarios ......................................................................................... 7
General Concepts: ........................................................................................................................... 8
  Defining an EFF Administrator User per node ............................................................................ 8
  Defining a non-root Linux account for installation and operation ............................................. 8
Setting Environment Variables to override default values prior to installation ............................. 9
Installation of the EFF Components .............................................................................................. 10
  Install help or usage: ................................................................................................................. 11
  Installing the Smart Licensing Tool: ......................................................................................... 13
  Installing the EFF Message Broker: ......................................................................................... 13
  Installing the System Administrator: ....................................................................................... 14
  Installing the System Monitor: ................................................................................................. 14
  Installing the ParStream Historian Database: .......................................................................... 15
Installing all of the EFF Linux components: ................................................................. 17
Upgrading to EFF version 1.1.0 from an existing version 1.0.x installation .................. 19
Configuring SSL certificate for secure connections ....................................................... 21
Starting and stopping the EFF Message Broker .......................................................... 21
Starting and stopping the EFF ParStream Historian Database .................................... 21
Starting and Stopping EFF System Administrator, EFF System Monitor and EFF DataFlow Editor .......................................................... 22
Connecting to the EFF System Components .............................................................. 23
Troubleshooting ........................................................................................................... 23
  Linux Firewall issues ................................................................................................. 23
  Proxy Server challenges and the EFF Message Broker ............................................ 23
EFF Message Broker Server Configuration via the server.json file ......................... 25
Obtaining documentation and submitting a service request ........................................ 29
Introduction

The Cisco Edge and Fog Fabric (EFF) allows you to create a reliable data communications messaging system on top of your data networking infrastructure that provides data delivery and allows you to rapidly deploy applications where needed that can be at the edge, fog or in the datacenter. The Edge and Fog Fabric is an open platform that allows for the addition of micro services or applications by anyone, allowing for unlimited capability and growth by adding software components that optimize the results of the application, system or outcome.

The EFF addresses the complexity of building an enterprise ready scalable data messaging system for one or many applications to reside upon. The EFF comes with a series of tools to manage the system, the EFF system administrator and the EFF system monitor.

Features and Functions

The system’s key capabilities include:

- Reusable micro services for collecting data from, and providing control over, devices and machines, as well as processing the data prior to delivery to its destination.
- Different options for reliable transport of data through the system, encompassing both batch and real-time streaming options.
- Flexible mechanisms for integration with IT systems, reporting, and analytics.
- An architectural framework to extend fog processing to multiple tiers: east west (fog to fog) and north south (hierarchical processing leveraging network topology).
- Easy-to-use GUI tools to simplify development, deployment, and operation for all aspects of the system.
- A pervasive control paradigm and flow of information back to micro services, devices and machines for management, control, optimization and specific actions.
- A completely open and polyglot system, where third parties can provide devices, processing storage, software modules, analytics, applications, or any combination thereof.

This is the technology that makes IoT approachable, and leads to much faster industry adoption of the vision of IoT.
## The Edge and Fog Fabric Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EFF Message Broker</strong></td>
<td>Provides reliable and flexible data delivery between any devices and micro services. The sources can be devices like sensors or other micro services and consumers can be micro services or user applications. The EFF Message Broker is a small footprint component working with other brokers to form a message bus.</td>
</tr>
<tr>
<td><strong>EFF Data Flow Editor</strong></td>
<td>Defines message paths between devices and micro services.</td>
</tr>
<tr>
<td><strong>EFF Data Flow Engine</strong></td>
<td>Executes message paths between devices and micro services. It is recommended to be installed adjacent to the EFF Message Broker in order to perform data transformation and input sources that not in the canonical data format of the system.</td>
</tr>
<tr>
<td><strong>EFF System Administrator</strong></td>
<td>Configures and manages the message broker and micro services.</td>
</tr>
<tr>
<td><strong>EFF System Monitor</strong></td>
<td>A standalone tool for operators to obtain real-time functional status of a deployed solution.</td>
</tr>
<tr>
<td><strong>Cisco ParStream (Historian Database)</strong></td>
<td>Purpose-built database for scale to handle the massive volumes and high velocity of IoT data as well as analytics at the Edge.</td>
</tr>
<tr>
<td><strong>EFF Tools Runtime Engine</strong></td>
<td>A standalone runtime tool for visualizing dashboards and driving EFF System Administrator, EFF Data Flow Engine and EFF System Monitor.</td>
</tr>
<tr>
<td><strong>Links</strong></td>
<td>DQL Link – DSA Query Language System Link – System Information ParStream Link – ParStream Historian Database</td>
</tr>
<tr>
<td><strong>Smart License Agent Tool for Nodes</strong></td>
<td>The Smart License Agent client that allows system users to manage license registration for Node Product IDs</td>
</tr>
<tr>
<td><strong>Smart License Agent Tool for Devices</strong></td>
<td>The Smart License Agent client that allows system users to manage license registration for Device Product IDs</td>
</tr>
</tbody>
</table>
## Hardware Requirements

<table>
<thead>
<tr>
<th>EFF Message Broker</th>
<th>Red Hat Linux 7.2, CentOS 7 or Ubuntu 16.04, 1GB RAM, Windows 2016 Server, 10 GB HD* - Recommended on the same system/VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFF Data Flow Engine</td>
<td>Automatically installs with EFF Message Broker and EFF Tools Runtime Engine. Access via a web browser</td>
</tr>
<tr>
<td>DQL Link</td>
<td>Project installs on the same system as the EFF Message Broker and EFF Tools Runtime Engine. Accessed via a web browser</td>
</tr>
<tr>
<td>System Link</td>
<td>Project installs on the same system as the EFF Message Broker and EFF Tools Runtime Engine. Accessed via a web browser</td>
</tr>
<tr>
<td>ParStream Link</td>
<td>Project installs on the same system as the EFF Message Broker and EFF Tools Runtime Engine. Accessed via a web browser</td>
</tr>
<tr>
<td>EFF Data Flow Editor</td>
<td>Automatically installs with EFF Message Broker and EFF Tools Runtime Engine. Access via a web browser</td>
</tr>
<tr>
<td>EFF System Administrator</td>
<td>Project installs on the same system as the EFF Message Broker and EFF Tools Runtime Engine. Accessed via a web browser</td>
</tr>
<tr>
<td>EFF System Monitor</td>
<td>Project installs on the same system as the EFF Message Broker and EFF Tools Runtime Engine. Accessed via a web browser</td>
</tr>
<tr>
<td>Cisco ParStream (Historian Database)</td>
<td>Red Hat Linux 7.2, CentOS 7 or Ubuntu 16.04, 6 CPU cores with 2GB RAM per core, 500 GB HD</td>
</tr>
<tr>
<td>EFF Tools Runtime Engine</td>
<td>Installs with EFF Message Broker</td>
</tr>
<tr>
<td>Smart License Agent Tool for Nodes and Smart License Agent Tool for Devices</td>
<td>Redhat Linux 7.2, CentOS 7 or Ubuntu 16.04, with 1GB RAM, 10 GB HD.</td>
</tr>
</tbody>
</table>

## EFF Components Protocols and Ports

The protocols and ports used by the EFF Broker and the EFF Historian Database. The port values are configurable during and after installation.

<table>
<thead>
<tr>
<th>TCP Port No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8080</td>
<td>Default http value for inbound connections to the Message Broker and Web access</td>
</tr>
<tr>
<td>8443</td>
<td>Default https value for inbound connections to the Message Broker and Web access</td>
</tr>
<tr>
<td>EFF Administrator Defined</td>
<td>Connection to ParStream Historian Server (when installed) from a local or remote ParStream DSLink. This includes the registrationPort and server(s) port in the server INI file.</td>
</tr>
</tbody>
</table>
Licensing installation and requests

This product uses the Smart License Agent Tool (for Nodes and Devices) to manage the corresponding smart licenses. After installation, refer to the IoT Data Connect - Edge Fog and Fabric 1.0 Smart License Agent User Guide.

Required Third Party Libraries for EFF Installation on Linux

The following third party libraries are needed for some of the EFF components. The exact versions listed were tested to work with the EFF.

For the ParStream Historian Database:

**CentOS 7, RHEL 7, Debian**

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>libicu</td>
<td>libicu-50.1.2-15.el7.x86_64</td>
</tr>
<tr>
<td>gpgme</td>
<td>gpgme-devel-1.3.2-5.el7.x86_64, gpgme-1.3.2-5.el7.x86_64</td>
</tr>
<tr>
<td>openssl</td>
<td>openssl-1.0.1e-60.el7_3.1.x86_64, openssl-devel-1.0.1e-60.el7_3.1.x86_64, openssl-1.0.1e-60.el7_3.1.x86_64</td>
</tr>
</tbody>
</table>

**Ubuntu 16.04**

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>libssl1.0.0</td>
<td>1.0.2g-1ubuntu4.6</td>
</tr>
<tr>
<td>libicu55</td>
<td>55.1-7ubuntu0.1</td>
</tr>
</tbody>
</table>

For the ParStream DSA Link and the Cisco Smart License Agent: Java 8 JRE or JDK

Secure Mode Operation

The can operate in Secure Mode to enhance the security features available for the EFF message broker and web server. Secure mode provides these following enhancements:

- HTTPS Strict Transport Security (HSTS), automatically redirecting inbound http connections to https for message broker and web traffic
- System dslink can not execute “system command”
- login page won’t allow browser to remember password
- prevents the pages from being embedded in iframes
• prevents the command action that allows shell exection by the System Link
Secure Mode is configured by indicating Y(es) during the installation of the message broker or
placing the hidden file “.secureMode” in the eff_server directory.

Using Secure Mode HSTS only affects inbound connections, outbound http and https
connections are still supported.

Basic component installation scenarios

The EFF has many components, allowing for a diverse manner of architecting a solution. While
there is no single deployment architecture, we will explain the basic deployment scenarios.

The first and simplest installation is a complete install on a single host. This scenarios allows for
development and testing, but is not typical for a production system.

The EFF architecture can be divided into five main building blocks and are typically on different
hosts. We can separate them as follows:
• EFF Smart License Agent(s) - this is the only node in the system that is required to
connect to Cisco.com, either directly or via the Smart License Satellite. The Smart
License Agent allows for license activation, revocation and renewals. Without license
activation or periodic communication over the Internet to Cisco.com, the EFF is out of
license compliance.
• EFF Message Broker with core DSLinks (DQL, System, ParStream and Dataflow engine).
The message broker is deployed on all nodes and is responsible for communications
between all components across the system.
• A System Administrator node, typically one per system. The is the administrative
console that allows for configuration and operation of the EFF System components. A
message broker is installed on this node in addition to the EFF System Administrator
project.
• A System Monitor node, also one per system. The EFF System Monitor allows operators
to view the connectivity and operations of the message brokers and DSLinks deployed
throught the system. The System Monitor is used as an operations console.¹

¹ The operation of the System Monitor acts as a subscription to the System link in every broker
for telemetry. If certain parts of the EFF system are bandwidth restricted, the use monitor rules
should be reduced.
• A Historian node, is deployed through a system to persist telemetry into a database. This is an add-on to a message broker. The ParStream DSLink is used to communicate between the message broker and the ParStream historian database.

The System Administrator and System Monitor use the message brokers for communications to all the nodes and system dslinks. Message broker to broker communications needs to be setup first before other tasks can be performed.

**General Concepts:**

**Defining an EFF Administrator User per node**

It is important to note that we do not define default username and passwords for EFF. The first user that is defined at install becomes the administrator of that node. After the install, using the System Administrator, the additional users that may be added. At least one user requires administrative privileges for that node.

**Defining a non-root Linux account for installation and operation**

As a Linux security best practice, it is recommend creating a non-root account for installing and operating the EFF. We suggest creating account “eff” for this purpose, and the following examples throughout the documentation will reference this name. The `adduser` command creates a new user “eff” and a new group “eff”.

To add a user, follow the instructions below for each support operating system:

**Redhat 7.2/Centos 7.2**

Adding the “eff” user account with a password and creating a home directory

```
$ sudo adduser eff -m -p <password>
```

**Ubuntu 16.04**

```
$ sudo adduser eff
Enter new UNIX password: <password>
Retype new UNIX password: <password>
Enter the new value, or press ENTER for the default
Full Name []: eff
Room Number []:
Work Phone []:
Home Phone []:
Other []:
```
For all operating systems:

As the host administrator (sudo), create the EFF root installation directory. Unless otherwise defined, this will be /opt/cisco/iotdc. Also change owner and group to “eff”.

$ sudo mkdir -p /opt/cisco/iotdc
$ sudo chown eff /opt/cisco/iotdc
$ sudo chgrp eff /opt/cisco/iotdc

Logging in as user “eff” from the current user

$ su eff

Place the EFF software image in the eff home directory.
The software should be downloaded from CCO at www.cisco.com under Support and Downloads.

Unzip the image

$ unzip EFF-1-1-0.zip

Change into unzipped folder

$ cd EFF-1-1-0

Setting Environment Variables to override default values prior to installation

Setting environment variables allows to override default value used during the installation script. A system installation will function properly by installing with default values, but there may be a need to change the root installation directory or broker socket ports because of deployment requirements.

In addition, by being able to define values, the installation can be scripted for multiple installations on many nodes.

For example, a sample script for installation:

$ export EFF_ROOT=/opt/cisco/iotdc
$ ./eff-linux broker

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFF_ROOT</td>
<td>should be set to the <em>absolute</em> path of the install root (i.e.</td>
<td>/opt/cisco/iotdc</td>
</tr>
</tbody>
</table>
### Installation of the EFF Components

The EFF installer eff-linux is a script that is designed to allow for interactive or non-interactive installation of the EFF components. The defaults allow for non-root users to operate the system.

All the examples below rely on default values and invocations refer to a fictitious $HOME being /home/userid and further assume, that the package EFF-1-1-0.zip has been unpacked inside that folder.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFF_GUI_LOGIN</td>
<td>set to the <em>name</em> of the EFF GUI Admin User</td>
<td>effAdmin</td>
</tr>
<tr>
<td>EFF_GUI_PHRASE</td>
<td>set to the <em>pass phrase</em> of the EFF GUI Admin User</td>
<td>Not set</td>
</tr>
<tr>
<td>EFF_BROKER_PRIV_KEY_PEM</td>
<td>overwrites the default value key.pem for certKeyName in server.json (i.e. with letsencrypt would suggest: /etc/letsencrypt/live/fully.qualified.domain.name/privkey.pem)</td>
<td></td>
</tr>
<tr>
<td>EFF_BROKER_FULL_CHAIN_PEM</td>
<td>overwrites the default value server.pem for certName in server.json (i.e. with letsencrypt would suggest: /etc/letsencrypt/live/fully.qualified.domain.name/fullchain.pem)</td>
<td></td>
</tr>
<tr>
<td>EFF_BROKER_SECURE_PORT</td>
<td>overwrite the default value for httpsPort in server.json</td>
<td>8443</td>
</tr>
<tr>
<td>EFF_BROKER_CLEARTEXT_PORT</td>
<td>overwrite the default value for port in server.json (disable this insecure access port by setting value 0 in production systems!)</td>
<td>8080</td>
</tr>
<tr>
<td>EFF_BROKER_IS_ALWAYS_OFFLINE</td>
<td>will overwrite the default value to set isAlwaysOffline true in server.json if set (isAlwaysOffline indicates that a server is expected to never have a full internet connection)</td>
<td>false</td>
</tr>
<tr>
<td>EFF_BROKER_WORKERS</td>
<td>may be set to positive integers in [1, 128] and should match the number of logical cpu cores as maximum and only if machine is dedicated to broker use and does not e.g. run local links or ParStream DB (if set and valid will overwrite the CPU core count derived one as value of the workers key in servers.json)</td>
<td>Maximum of: 1 and CPU Cores / 3</td>
</tr>
<tr>
<td>EFF_INSTALL_LOGS</td>
<td>set to the <em>absolute</em> path of a folder for install logs (default: case logs subfolder of EFF-1.0 e.g. $HOME/EFF-1.0/logs if EFF-1.0.zip unpacked in $HOME folder) (e.g. needed in case the unpacked install components are stored on a read-only medium)</td>
<td></td>
</tr>
<tr>
<td>EFF_DEBUG</td>
<td>the installer is run with more explicit warn and fail early flags</td>
<td></td>
</tr>
</tbody>
</table>
Install help or usage:

Executing `./eff-linux help` displays a summary help screen.

Executing `./eff-linux help` displays an extended help screen.

INFO: Interactive mode enabled (reason default)

Sample usage(s):

- `eff-linux install`  
  # -> install all components (dart, broker, admin, monitor, license, and parstream)

- `eff-linux upgrade`  
  # -> upgrades all platform components (broker, admin, monitor, and parstream)

- `eff-linux broker`  
  # -> installs broker, data flow editor and dsa run-time

- `eff-linux dart`  
  # -> installs dart run-time

- `eff-linux admin`  
  # -> installs admin

- `eff-linux install license`  
  # -> installs smart license agent

- `eff-linux parstream`  
  # -> installs parstream historian database

- `eff-linux verify`  
  # -> report current settings and SHA512 checksums of components

- `eff-linux verify dart`  
  # -> calculate SHA512 checksum of dart run-time

- `eff-linux version`  
  # -> report version banner of this tool

- `eff-linux [help]`  
  # -> this help screen / usage info.

Notes:

Installing admin and monitor will also install Dart VM and Broker run-time.

Environment Variables:

- `EFF_ROOT` should be set to the *absolute* path of the install root (default: `/opt/cisco/iotdc`)
  (i.e. folders `eff_server`, `parstream`, and `dart-sdk` will be created below that path)
  Current value is: `"/opt/cisco/iotdc"` (without the enclosing single quotes)

- `EFF_GUI_LOGIN` may be set to the *name* of the EFF GUI Admin User (default: `effAdmin`)
  Current value is: `<UNSET>`

- `EFF_GUI_PHRASE` may be set to the *pass phrase* of the EFF GUI Admin User (default: not set)
  (This variable will be ignored for now in interactive install sessions)
  Current value is: `<UNSET>`
EFF_BROKER_PRIV_KEY_PEM overwrites the default value key.pem for certKeyName in server.json (i.e. with letsencrypt would suggest: /etc/letsencrypt/live/fully.qualified.domain.name/privkey.pem)
Current value is: <UNSET>

EFF_BROKER_FULL_CHAIN_PEM overwrites the default value cert.pem for certName in server.json (i.e. with letsencrypt would suggest: /etc/letsencrypt/live/fully.qualified.domain.name/fullchain.pem)
Current value is: <UNSET>

EFF_BROKER_SECURE_PORT will overwrite the default value of 8443 for httpsPort in server.json
Current value is: <UNSET>

EFF_BROKER_CLEARTEXT_PORT will overwrite the default value of 8080 for port in server.json (disable this insecure access port by setting value 0 in production systems!)
Current value is: <UNSET>

EFF_BROKER_IS_ALWAYS_OFFLINE will overwrite the default value to set isAlwaysOffline true in server.json (isAlwaysOffline indicates that a server is expected to never have a full internet connection)
Current value is: <UNSET>

EFF_BROKER_WORKERS may be set to positive integers in \[1, 128\] and should match the number of logical cpu cores as maximum and only if machine is dedicated to broker use and does not e.g. run local links or ParStream DB (if set and valid will overwrite the CPU core count derived one as value of the workers key in servers.json)
Current value is: <UNSET>

EFF_INSTALL_LOGS may be set to the *absolute* path of a folder for install logs (default: /home/userid/EFF-1-1-0) (e.g. needed in case the unpacked install components are stored on a read-only medium)
Current value is: <UNSET>

EFF_UNATTENDED may be set to fast enable unattended operation solely controlled by eff.json file content. Setting this to anything else than an empty value is equivalent to setting INTERACTIVE to false in eff.json

EFF_DEBUG: For execution in debug mode, please set EFF_DEBUG environment variable to nonempty value

Copyright:
(c) 2017 Cisco and/or its affiliates. All rights reserved.

Executing `.*/eff-linux version` displays the version.

Executing `.*/eff-linux env` reports the effective environment variable values and their source.
Installing the Smart Licensing Tool:
In this example no global variables are set, all defaults are used.

Typing ./eff-linux license

INFO: Interactive mode enabled (reason default)
INFO: The component EFF Smart License Agent has been installed

Installing the EFF Message Broker:
In this example no global variables are set, all defaults are used.

This installs the EFF Message Broker and the DQL, System, ParStream and Dataflow engine DSLinks.

Executing ./eff-linux broker

INFO: Interactive mode enabled (reason default)
INFO: Dart-Broker server configuration template copied successfully from /home/userid/EFF-1-1-0/Components/server.json
INTERACTION: Reconfigure the Dart-Broker server configuration? [y/n]: y
INFO: Dart-Broker server configuration reconfiguration requested, continuing ...
INFO: Set server json value of certName from EFF_BROKER_FULL_CHAIN_PEM to cert.pem
INFO: Set server json value of certKeyName from EFF_BROKER_PRIV_KEY_PEM to key.pem
INFO: Set server json value of httpsPort from EFF_BROKER_SECURE_PORT 8443
INFO: Set server json value of isAlwaysOffline from EFF_BROKER_IS_ALWAYS_OFFLINE to false
INFO: Set server json value of port from EFF_BROKER_CLEARTEXT_PORT to 8080
INFO: Set server json value of workers from EFF_BROKER_WORKERS to 1
INFO: Dart-Broker server configuration updated with path
INFO: Installation created secure by default setup, good.
INTERACTION: Keep blocking unsecured creation of upstream connections over HTTP using the EFF Dart installation? [Y/n]: y
INFO: Setup kept secure as installed by default, good.
INFO: Dart-Broker user configuration template copied ...
INTERACTION: Perform Dart-Broker custom user configuration now? [y/n]: y
INFO: Dart-Broker custom user configuration requested, continuing ...
INTERACTION: EFF GUI Admin login is (effAdmin). Change? [y/N]:

INTERACTION: Enter Password:
INTERACTION: Enter Password (verify):
INFO: Map the given login credentials into the user configuration and grant admin rights ...
Installing the System Administrator:
In this example no global variables are set, all defaults are used.

The System Administrator is an option component that can be added to the broker for managing the system. It requires that the broker previously has been installed and configured to properly function and thus will trigger install of broker otherwise.

Executing ./eff-linux admin

INFO: Interactive mode enabled (reason default)
INFO: The component EFF System Administrator has been installed

Installing the System Monitor:
In this example no global variables are set, all defaults are used.

Typing ./eff-linux monitor

Since the System Monitor is envisioned to operate on an operations console, the installation takes the necessary steps of installing the message broker and system monitor project as a bundle. So in contrast to above admin install, the below sample starts with an empty target folder.

INFO: Interactive mode enabled (reason default)
INFO: Sub-task install Dart Broker (required by Monitor) ...
INFO: Sub-task install Dart VM (required by Dart broker) ...
INTERACTION: Installed platform dart run-time. Check that the VM can execute and expose it's version? [Y/n]:
INFO: dart VM check succeeded with Dart VM version: 1.21.1 (Fri Jan 13 02:01:35 2017) on "linux_x64"
INFO: Sub-task install Dart VM (required by Dart broker) succeeded
INFO: Dart-Broker server configuration template copied successfully from /home/userid/EFF-1-1-0/Components/server.json
INTERACTION: Reconfigure the Dart-Broker server configuration? [y/n]: y
INFO: Dart-Broker server configuration reconfiguration requested, continuing ...
INFO: Set server json value of certName from EFF_BROKER_FULL_CHAIN_PEM to cert.pem
INFO: Set server json value of certKeyName from EFF_BROKER_PRIV_KEY_PEM to key.pem
INFO: Set server json value of httpsPort from EFF_BROKER_SECURE_PORT 8443
INFO: Set server json value of port from EFF_BROKER_CLEARTEXT_PORT to 8080
INFO: Set server json value of workers from EFF_BROKER_WORKERS to 1
INFO: Dart-Broker server configuration updated with path
INFO: Installation created secure by default setup, good.
INTERACTION: Keep blocking unsecured creation of upstream connections over HTTP using the EFF Dart
installation? [Y/n]:
INFO: Setup kept secure as installed by default, good.
INFO: Dart-Broker user configuration template copied ...
INTERACTION: Perform Dart-Broker custom user configuration now? [y/n]: y
INFO: Dart-Broker custom user configuration requested, continuing ...
INTERACTION: EFF GUI Admin login is (effAdmin). Change? [y/N]:

INTERACTION: Enter Password:
INTERACTION: Enter Password (verify):
INFO: Map the given login credentials into the user configuration and grant admin rights ...
INFO: Creation of EFF GUI Admin User effAdmin succeeded with ["User 'effAdmin' was successfully
created."]
INFO: Grant admin rights for EFF GUI Admin User effAdmin succeeded with ["User 'effAdmin' was
successfully granted superuser."]
INFO: Users entry 1/1
REPORT: Username is: effAdmin
REPORT: Admin rights granted: true
REPORT: Passphrase in PBKDF2 format:
044gXjOfke0vYWPAu+86fthex1YNPyjOoDqY1c0r0JvO=6tpP1leymdO0d/Mh+IyvHUNme/AxX8nNjIWMRBO/RrQVQ=
REPORT: DSA Version: 1.1.2', 'DGLux5 Build: r7712', 'DGLux Server Build: 1156', 'Verifying Environment...', 'Verifying Configuration...'
INFO: Installed broker and run-time. Check that the Dart-Broker can execute and expose it's
version? [Y/n]:
INFO: Dart-Broker check succeeded with ['DSA Version: 1.1.2', 'DGLux5 Build: r7712', 'DGLux Server
Build: 1156', 'Verifying Environment...', 'Verifying Configuration...']
INFO: Sub-task install Dart Broker (required by Monitor) succeeded
INFO: The component EFF System Monitor has been installed

Installing the ParStream Historian Database:

Important note on ParStream Historian installation:
- Operational data, e.g. partitions, configuration files or journals should never be stored in
the ParStream installation directory.

In this example no global variables are set, all defaults are used.

This installs the ParStream Historian Database.

Executing ./eff-linux parstream
This will install the ParStream Historian Database to the folder specified in the environment variable `EFF_ROOT`. In the following example `EFF_ROOT` was set to `/home/eff/opt`.

```
INFO: Interactive mode enabled (reason default)
INFO: Check of platform support - passed. (ok)
INFO: Check of cpu core count - passed. (ok)
INFO: Check of soft limit for open files - passed. (ok)
INTERACTION: Installed historian/ParStream. Check that the parstream-server can execute and expose it's version? [Y/n]:
INFO: For future execution of the historian/ParStream server command is
/home/eff/opt/parstream/bin/parstream-server
INFO: Executing check that the parstream-server can execute and expose it's version
```

```
DEBUG: [2017-10-20T15:09:11]:unknown-000000:PROT-77001: ****** Starting Cisco ParStream Server 5.0.0 20171005T124056Z Release
DEBUG: /home/eff/opt/parstream/bin/parstream-server version: 5.0.0 built: 20171005T124056Z (7fe2df5df15cb08b4cbe9a6e05e89459c6662a)
```

```
INFO: The component EFF ParStream Historian Database has been installed
```

It is recommended to set the environment PARSTREAM_HOME, LD_LIBRARY_PATH and PATH as suggested by the install script:

```
export PARSTREAM_HOME=/home/eff/opt/parstream
export LD_LIBRARY_PATH=$PARSTREAM_HOME/lib:$LD_LIBRARY_PATH
export PATH=$PATH:$PARSTREAM_HOME/bin
```

After installing ParStream, the installer will ask you to verify the installation. If everything was installed successfully, it will print the ParStream version and exit.

In the ParStream installation folder (In this example found in `/home/eff/opt/parstream`) an examples directory exists. These examples provide you with basic ParStream configuration files. In each example’s directory a conf directory exists. In this directory you’ll find the parstream.ini file. This file provides the configuration parameters the ParStream Historian Database will read on startup.

Please refer to section “Setting Environment Variables to override default values prior to installation” for more information about environment variables.

We use the user `eff` in this example.
In order to execute a specific example navigate to the example’s directory. Here you’ll find the shell script `run_cluster.sh`. When invoking this shell script the respective example will be executed.

For more details regarding the configuration of ParStream and further details, please refer to the ParStream documentation accompanying this document. The additional documentation files accompany this document.

### Installing all of the EFF Linux components:

In this example no global variables are set, all defaults are used.

For a complete installation of all the components on a host system, you can use this command option. We do not envision production systems will be designed with all the components on a single system, there are circumstances for development testing and learning environments that desire a quicker installation with one command.

This command installs the EFF Smart Licensing Tool, EFF Message Broker, EFF System Administrator, EFF System Monitor, EFF DataFlow Editor and the EFF ParStream Historian Database.

#### Executing `/eff-linux install`

```
INFO: Interactive mode enabled (reason default)
INTERACTION: Installed platform dart run-time. Check that the VM can execute and expose it's version? [Y/n]: 
INFO: dart VM check succeeded with Dart version: 1.21.1 (Fri Jan 13 02:01:35 2017) on "linux_x64"
INFO: The component Dart VM has been installed
INFO: Dart-Broker server configuration template copied successfully from /home/eff/EFF-1-1-0/Components/server.json
INTERACTION: Reconfigure the Dart-Broker server configuration? [y/n]: y
INFO: Dart-Broker server configuration reconfiguration requested, continuing ...
INFO: Set server json value of certName from EFF_BROKER_FULL_CHAIN_PEM to cert.pem
INFO: Set server json value of certKeyName from EFF_BROKER_PRIV_KEY_PEM to key.pem
INFO: Set server json value of httpsPort from EFF_BROKER_SECURE_PORT 8443
INFO: Set server json value of isAlwaysOffline from EFF_BROKER_IS_ALWAYS_OFFLINE to false
INFO: Set server json value of port from EFF_BROKER_CLEARTEXT_PORT to 8080
INFO: Set server json value of workers from EFF_BROKER_WORKERS to 1
INFO: Dart-Broker server configuration updated with path
INFO: Installation created secure by default setup, good.
INTERACTION: Keep blocking unsecured creation of upstream connections over HTTP using the EFF Dart installation? [Y/n]: 
INFO: Setup kept secure as installed by default, good.
INTERACTION: Perform Dart-Broker custom user configuration now? [y/n]: y
INFO: Dart-Broker custom user configuration requested, continuing ...
INTERACTION: EFF GUI Admin login is (effAdmin). Change? [y/N]: 
INTERACTION: Enter Password:
INTERACTION: Enter Password (verify):
INFO: Map the given login credentials into the user configuration and grant admin rights ...
```
INFO: Creation of EFF GUI Admin User effAdmin succeeded with ["User 'effAdmin' was successfully created."]
INFO: Grant admin rights for EFF GUI Admin User effAdmin succeeded with ["User 'effAdmin' was successfully granted superuser."]
INFO: Users entry 1/1
REPORT: Username is: effAdmin
REPORT: Admin rights granted: true
REPORT: Passphrase in PBKDF2 format: 044aCUjVhlpjssHxLd6NVM641b9ch64H0vi2KzN2xXyc8B8=5oD6/s2/6sou3uP6/pAvS1py0ZxthmaQ2/xjbJKRUu018=
INTERACTION: Installed broker and run-time. Check that the Dart-Broker can execute and expose it's version? [Y/n]:
INFO: Dart-Broker check succeeded with ['DSA Version: 1.1.2', 'DGLux5 Build: r7712', 'DGLux Server Build: 1156', 'Verifying Environment...', 'Verifying Configuration...']
INFO: The component EFF Message Broker has been installed
INFO: The component EFF System Administrator has been installed
INFO: The component EFF System Monitor has been installed
INFO: The component EFF Smart License Agent has been installed
INFO: Check of platform support - passed. (ok)
INFO: Check of cpu core count - passed. (ok)
INFO: Check of soft limit for open files - passed. (ok)
INTERACTION: Installed historian/ParStream. Check that the parstream-server can execute and expose it's version? [Y/n]:
INFO: Check that the parstream-server can execute and expose it's version requested
INFO: For future execution of the historian/ParStream possible environment settings are:
INFO: LD_LIBRARY_PATH=/home/eff/opt/parstream/lib
INFO: For future execution of the historian/ParStream server command is
INFO: Executing check that the parstream-server can execute and expose it's version
DEBUG: [2017-10-20T16:17:15]:unknown-000000:PROT-77001: ****** Starting Cisco ParStream Server 5.0.0
DEBUG: [2017-10-20T16:17:15]:unknown-000000:PROT-77083: **** cmdlineargs: --version
INFO: The component EFF ParStream Historian Database has been installed
Upgrading to EFF version 1.1.0 from an existing version 1.0.x installation

If an existing version 1.0.0 or 1.0.1 is installed and running on a Linux server it is possible to upgrade to the version 1.1.0 without a fresh install. It is highly recommended performing a backup of the host prior to proceeding with the upgrade.

Assuming that the EFF version 1.0.x is already installed and running. Here are the pre-requisites to upgrading:
- Download the EFF 1.1.0 package
- Stop the running message broker (see below)
- Stop the running ParStream Historian Database (see below)

Important notes on ParStream Historian installation:

- Operational data, e.g. partitions, configuration files or journals should never be stored in the ParStream installation directory. When upgrading an existing EFF installation, we may decide to replace the ParStream installation folder by the new version. When upgrading from a previous version of EFF (1.0.0 or 1.0.1) to EFF 1.1.0, the installer will create a backup folder containing all files and directories in the existing ParStream installation directory.
- After upgrading to a new version of ParStream, please make sure that PARSTREAM_HOME and LD_LIBRARYPATH are set appropriately. Refer to the ParStream manual, accompanying this document, for more information.
- [Specific usage] If you are using BLOB columns in your existing installation, please update all ParStream DSA Links before updating the ParStream database server.

Perform the following steps to upgrade:

Prepare upgrade like a fresh install and unpack the EFF-1-1-0.zip accordingly and change current working directory to EFF-1-1-0 folder all as described above in the install section.

$> unzip -q EFF-1-1-0.zip
$> cd EFF-1-1-0

Set environment variable to the same destination as the current version 1.0.x installation, for example:

echo EFF_ROOT=/opt/cisco/iotdc

Stop the broker if running:
Finally upgrade your EFF installation to 1.1.0 with the following command and follow the instructions

```
./eff-linux upgrade
```

After a successful upgrade, restart the message broker (see below).
Configuring SSL certificate for secure connections

A necessary step for the message broker and web server to allow incoming secure connections is to properly install the SSL certificates and define a non-null (or not empty value) value for the certificate password in the server.json file.

The EFF system ships with a self-signed SSL certificate and certificate private key. It is optional to install user certificates. The placement of the certificate files are in the $EFF_ROOT/eff_server/certs folder.

This is accomplished during the broker installation or editing the $EFF_ROOT/eff_server/server.json file and adding a non-null value to the “certPassword” field. Without this value, the SSL functionality will not operate.

Starting and stopping the EFF Message Broker

This step starts the EFF Message Broker, EFF Data Flow Engine and Editor and the DSLinks that might were installed.
Note: Assuming the default installation path of /opt/cisco/iotdc/eff_server in the following examples.

Starting and Stopping EFF Message Broker:
To start the EFF Message Broker, EFF Data Flow Engine and Editor execute the following:
/opt/cisco/iotdc/dart-sdk/bin/dart /opt/cisco/iotdc/eff_server/bin/daemon.dart start

To stop the EFF Message Broker, EFF Data Flow Engine and Editor execute the following:
/opt/cisco/iotdc/dart-sdk/bin/dart /opt/cisco/iotdc/eff_server/bin/daemon.dart stop

If the message broker ports are in the range of 1-1024, the Linux operating system will require sudo or root priveledges to start the application.

$sudo /opt/cisco/iotdc/dart-sdk/bin/dart /opt/cisco/iotdc/eff_server/bin/daemon.dart start
$sudo /opt/cisco/iotdc/dart-sdk/bin/dart /opt/cisco/iotdc/eff_server/bin/daemon.dart stop

Starting and stopping the EFF ParStream Historian Database

4 Note that this self-signed certificate will present a browser security exception. To eliminate the security warning, a valid certificate should be installed.
To start the ParStream Historian Database without the `run_cluster.sh` script, navigate into one of the example’s directories. If you want to follow this example please navigate to the `noauthentication` directory (Found here `/home/eff/opt/parstream/examples/noauthentication`, if installed as user `eff`).

In each example directory a `conf` folder is located. Within the `conf` folder the `parstrem.ini` file exists. The configuration of the ParStream Historian Database cluster is also configured by the `parstrem.ini` file. Each cluster node has its own section in the file. The `noauthentication` example configures a one node cluster.

For this purpose the section `server.first` is specified in the `parstrem.ini` file. In order to start the cluster node execute the following command in the `noauthentication` directory:\(^5\):

```
$PARSTREAM_HOME/bin/parstream-server first
```

This will start the ParStream Historian Database with the global parameters found in the `parstrem.ini` file. The node specific parameters are read from the node’s ini file section (In this case `server.first`).

After the cluster’s initialization phase it will start to listen on port 9042 for incoming client connections.

In order to shutdown the ParStream Historian Database execute the following command:\(^6\):

```
$PARSTREAM_HOME/bin/pnc -p 9042
```

This will establish a connection the ParStream Historian Database server listening on port 9042. Issue the following commend when command prompt `Cisco ParStream=>` appears:

```
Cisco ParStream=> ALTER SYSTEM CLUSTER SHUTDOWN;
```

This will shut the cluster down. For more information about configuring, starting and stopping ParStream Historian Database clusters refer to the ParStream accompanying the package.

---

Starting and Stopping EFF System Administrator, EFF System Monitor and EFF DataFlow Editor

\(^5\) If the command fails, make sure that all environment variables are set as described.

\(^6\) Make sure that the ParStream Historian Database server is still running and listening on port 9042.
These components are available when the EFF Message Broker is running. No other steps are necessary to start these tools.

Connecting to the EFF System Components

All the EFF tools require the administrator user and login for access.

<table>
<thead>
<tr>
<th>Application</th>
<th>Insecure Port (if supported)</th>
<th>Secure Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFF System Administrator</td>
<td>http://[Server IP Address]/eff-admin</td>
<td>https://[Server IP Address]/eff-admin</td>
</tr>
<tr>
<td>EFF System Monitor</td>
<td>http://[Server IP Address]/eff-monitor</td>
<td>https://[Server IP Address]/eff-monitor</td>
</tr>
<tr>
<td>EFF Data Flow Editor</td>
<td>http://[Server IP Address]/dataflow.html</td>
<td>https://[Server IP Address]/dataflow.html</td>
</tr>
</tbody>
</table>

Troubleshooting

Linux Firewall issues

Redhat and CentOS initially is configured by default with the firewall service turned on and blocks all incoming connections. It is necessary to read the operating system guide to turn off or allow only the known service ports for the EFF connections. But the command below can be used for troubleshooting or staging. The proper configuration needs to be defined by the host administrator.\(^8\)

To turn off the firewall on Redhat/CentOS/ the following command can be executed:

```
$ sudo service firewalld stop
```

Proxy Server challenges and the EFF Message Broker

\(^7\) Note if the server configuration is using Safe Mode, income connections will automatically redirect to the secure port if configured. If no secure port is defined or certificates and passwords are not properly configured, the secure port will not accept connections and therefore in Safe Mode the access to the system is unavailable.

\(^8\) The firewall service will restart on next reboot unless the administrator modifies the default configuration.
In some environments it might be necessary to define a proxy server to access the Internet due to security restrictions. The EFF message broker uses a localhost communication to connect to the DSLinks on the same host and usually any proxy server configuration inhibits some of this functionality from functioning properly.

We have observed in the System Administrator that some DSLinks connect to the message broker, while others do not if there is a proxy server configured.

In order to successfully connect to all the DSLinks it stopping the message broker be necessary, remove the proxy settings and start again the message broker. For example:

- Stop the message broker with `/opt/cisco/iotdc/eff_server/bin/daemon.sh stop`
- Remove the proxy server settings in the environment or system configuration
- Start the message broker with `/opt/cisco/iotdc/eff_server/bin/daemon.sh start`
EFF Message Broker Server Configuration via the server.json file

Example server.json configuration file located in the $EFF_ROOT/eff_server folder.

```json
{
    "allowAllLinks": true,
    "allowBrowserCaching": false,
    "allowPasswordChanges": true,
    "alternativeBrokerUrl": null,
    "authType": "file",
    "broadcast": false,
    "brokerName": "broker-",
    "certKeyName": "key.pem",
    "certName": "cert.pem",
    "certPassword": "",
    "corsProxyRules": "",
    "dartRuntimeManagerVmFlags": [],
    "debug": false,
    "defaultPermission": null,
    "disableFileSecurity": false,
    "disabledLinks": [],
    "distributionUrl": "NO",
    "downstreamName": "downstream",
    "enableCertificateGeneration": true,
    "enableGid": false,
    "enableIPv6": false,
    "enableSingleSignOnServer": false,
    "enableUptimeChecker": true,
    "formatDg5": false,
    "guestLoginRedirectPath": "/assets/",
    "hooks": {},
    "host": "0.0.0.0",
    "httpPathClassification": {},
    "httpsPort": 8443,
    "isAlwaysOffline": false,
    "javaRuntimeManagerVmFlags": [],
    "keepCustomAssets": true,
    "linkConfig": {},
    "linkManagerEnvironment": {},
    "linkRepositoryUrl": "https://dsa.s3.amazonaws.com/links/links.json",
    "logRotationInterval": 0,
    "loggers": [],
    "loginRedirectPath": "/",
    "observe": false,
    "passwordHasherIterations": 1000,
    "passwordHasherKeyLength": 32,
    "port": 8080,
    "proxies": {},
    "quarantine": false,
    "runBrokerInMain": true,
    "runPortChecks": true,
    "serverLogLevel": "INFO",
    "serverVmFlags": [],
    "ssoProviderUrl": null,
    "static": {
        "/.well-known": "/opt/cisco/iotdc/eff_server/.well-known"
    },
    "storageDriver": "simple",
    "timeHttpRequests": false,
    "twoFactorAuth": "none",
    "updateInterval": 200,
    "upstream": {},
    "uptimeCheckUrl": null,
}"
```
In below table the default values are those, that are assumed by the server, if the key is not present in the server.json

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug</td>
<td>Enable/Disable Debugging Mode</td>
<td>false</td>
<td>For production site, this should always be false, debug:true may result in memory leak and bugs.</td>
</tr>
<tr>
<td>port</td>
<td>HTTP Port to listen on. If this is less than or equal to 0, then the server does not listen on any port for HTTP.</td>
<td>8080</td>
<td>At least one of port or httpsPort must have a valid port number assigned.</td>
</tr>
<tr>
<td>httpsPort</td>
<td>HTTPS port to listen on. If this is less than or equal to 0, and/or certName or certPassword is not provided, then the server does not listen on any port for HTTPS. Ensure that if you install a custom certificate, you fill in the certName, certKeyName and certPassword fields.</td>
<td>8443</td>
<td>At least one of port or httpsPort must have a valid port number assigned.</td>
</tr>
<tr>
<td>certName</td>
<td>SSL certificate file name. Leave blank to disable HTTPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>certPassword</td>
<td>SSL certificate password. Set to null to disable HTTPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>certKeyName</td>
<td>SSL private key file name. Leave blank to disable HTTPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disableFileSecurity</td>
<td>When this value is true, then any user can access any file. When this is false, file permissions are checked.</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>broadcast</td>
<td>When this value is true, the server’s broker is broadcast to the local network for discovery by other machines. When this value is false, the broadcast service is not enabled.</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>workers</td>
<td>Number of Server Workers. For low end devices, this should stay at 1. For large machines, this can be set up to a</td>
<td></td>
<td>For single-core machines, this is 1, for other devices,</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>maximum of 128.</td>
<td>It is recommended to not exceed the number of logical processors on your</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>static</td>
<td>Configures a static directory mapping. This is used to serve files and</td>
<td>{&quot;/static&quot;: &quot;srv/http/static&quot;}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>directories on the server. Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{&quot;/static&quot;: &quot;srv/http/static&quot;}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>defaultPerm</td>
<td>Default permission setting for the root node. When this value is null,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>permissions are disabled, and everything has the config permission.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>allowAllLinks</td>
<td>When the value is true, all incoming DSLink connections will be accepted</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to /downstream. When the value is false, an incoming DSLink without</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>proper authentication will be rejected unless quarantine is enabled.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quarantine</td>
<td>** This setting has no effect when allowAllLinks is true **</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When the value is true, a new incoming DSLink without a token will be</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>put in /sys/quarantine. A quarantined DSLink can only work as a responder.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use the /sys/quarantine/authorize to move a quarantined DSlink to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/downstream.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isAlwaysOffline</td>
<td>Indicates that a server is expected to never have a full internet</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>useDartRuntimeManager</td>
<td>When the value is true, the Dart Runtime Manager is used for Dart DSLinks.</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Dart runtime manager reduces resource consumption by merging Dart</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSLinks into a single process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>useJavaRuntimeManager</td>
<td>When the value is true, the Java Runtime Manager is used for Java DSLinks.</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Java runtime manager reduces resource consumption by merging Java</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSLinks into a single process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>guestLoginRedirectPath</td>
<td>Determines the URI that a user is redirected to when login is complete.</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>authType</td>
<td>Determines the authentication provider to use.</td>
<td>file</td>
<td></td>
</tr>
<tr>
<td>twoFactorAuth</td>
<td>Determines the two factor authentication provider to use.</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Supported Two-Factor Authentication Providers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• none: Don't enable two factor authentication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• duo: Duo Two-Factor Authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>enableIPv6</td>
<td>Toggles support for IPv6 connections</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>keepCustomAssets</td>
<td>When the value is true, custom assets in www/assets are kept upon updating EFF Server.</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>formatDg5</td>
<td>When this value is true, eff client will save dg5 in a formatted and json with key sorted, makes it easy to track changes.</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
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:


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.