FixedIT Data Agent

This Agent allows the user to run Telegraf in the camera using any number of configuration files. The configuration files are used to define the behavior of Telegraf and can be created and specified by the user. They can also be changed at runtime.

Telegraf can be used for a wide range of tasks, such as sampling, polling, pushing and processing data. The configuration files bundled with the application (i.e. the files that are used by default if no custom files are specified by the user) sample system metrics in the camera and push them to another server (e.g. InfluxDB).

Table of contents:

- * 0.0.1

Installing the application

The FixedIT Data Agent application is built for both `armv7hf` and `aarch64` architectures. The correct version should be installed depending on each device's architecture. If you are unsure which architecture your device has, see the `QUICKSTART_GUIDE.pdf` for further instructions.

Activating the license

This application requires a license to run. After installation, the application can be turned on, but if the license is not activated, it will not perform any tasks.

The license can be activated either by uploading an offline license file or by performing an online activation using a license code. For step-by-step instructions, see the `QUICKSTART_GUIDE.pdf` file.

User interface

The application is available in two versions:

Version with Web UI: This version includes a web-based user interface that allows you to:

- View Telegraf logs and status in real-time
- Upload, manage, and delete configuration files
- Monitor the application's health and status

Version without Web UI: This version runs without a web interface and is typically configured through:

- The FixedIT Installer Agent
- Direct parameter configuration in the camera's application settings
- SSH access for file management

Both versions output all standard output and error output from the Telegraf subprocess to the system log at predetermined intervals. The version with a web user interface also outputs this information in real-time to the web UI.

Navigating the UI

The version of the application that includes a web UI will show an "Open" button in the camera's UI, as shown below.

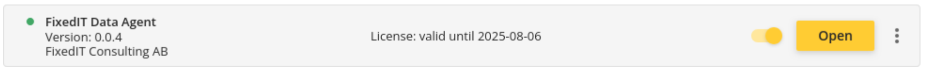
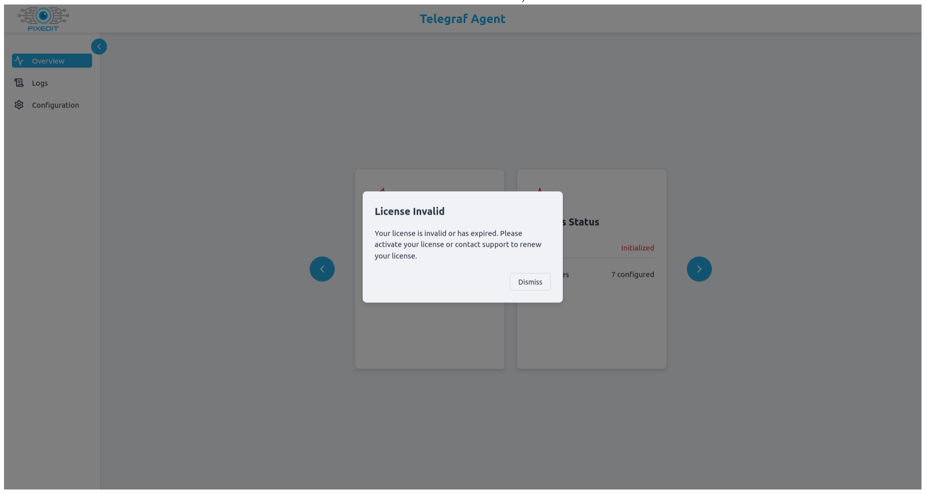

FixedIT Data Agent
Version: 0.0.4
FixedIT Consulting AB
License: valid until 2025-08-06

Figure 1: FixedIT Data Agent UI Open button

Clicking this button leads to the application's UI. The UI can also be accessed at `https://<CAMERA_IP>/local/FixeditDataAgent/index.html#/overview`.

Note that if the license is not activated, the UI will not be accessible:



License Invalid

Your license is invalid or has expired. Please activate your license or contact support to renew your license.

To the very left, there is a navigation tab showing all the available pages: `Overview`, `Logs` and `Configuration`.

Overview page

The default page that opens when accessing the UI from the Open button is the `Overview`.

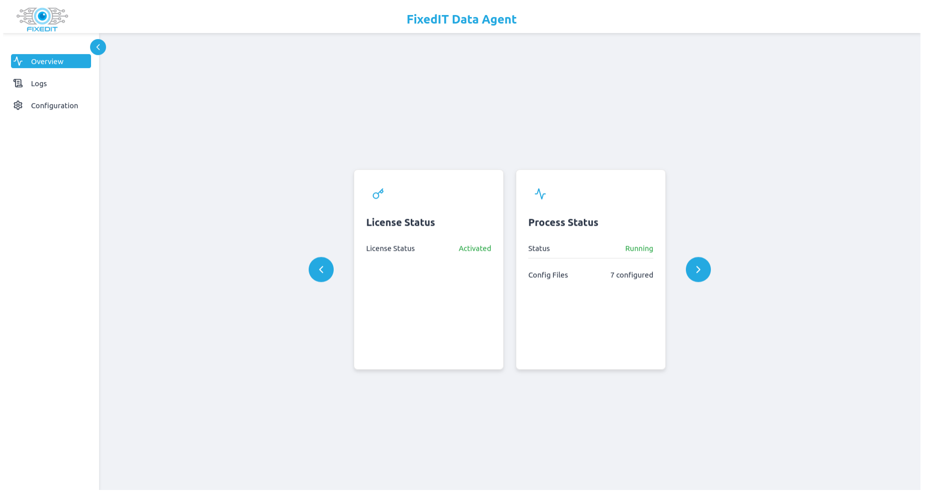


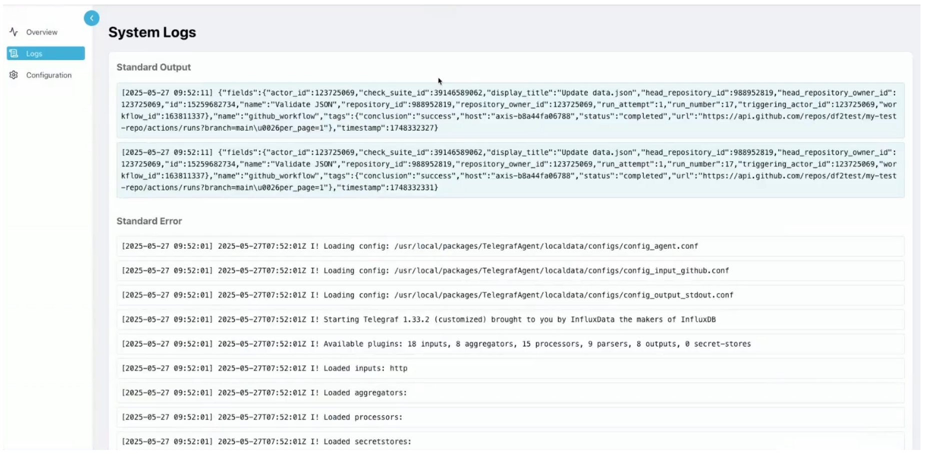
Figure 2: Overview page for FixedIT Data Agent

The box on the left shows the License Status. For instructions on how to install a license, see Activating the license.

The box on the right shows the process status. This indicates the status of the Telegraf process. If everything is running correctly, the status will show as `Running`.

Logs page

The `Logs` page shows the logs from the Telegraf process. The page is automatically updated with the latest logs, where the newest logs will show at the bottom. Errors are indicated with a pink background, warnings are indicated with a yellow background. `stdout` output is displayed first, then `stderr`.



System Logs

Standard Output

```
[2025-05-27 09:52:11] {"fields":{"actor_id":123725069,"check_suite_id":39146589062,"display_title":"Update data.json","head_repository_id":988952819,"head_repository_owner_id":
123725069,"id":15259682734,"name":"Validate JSON","repository_id":988952819,"repository_owner_id":123725069,"run_attempt":1,"run_number":17,"triggering_actor_id":123725069,"wor
kflow_id":163811337},"name":"github_workflow","tags":{"conclusion":"success","host":"axis-b8a44fa06788","status":"completed","url":"https://api.github.com/repos/df2test/my-test
-repo/actions/runs?branch=main\u0026per_page=1"},"timestamp":1748332327}

[2025-05-27 09:52:11] {"fields":{"actor_id":123725069,"check_suite_id":39146589062,"display_title":"Update data.json","head_repository_id":988952819,"head_repository_owner_id":
123725069,"id":15259682734,"name":"Validate JSON","repository_id":988952819,"repository_owner_id":123725069,"run_attempt":1,"run_number":17,"triggering_actor_id":123725069,"wor
kflow_id":163811337},"name":"github_workflow","tags":{"conclusion":"success","host":"axis-b8a44fa06788","status":"completed","url":"https://api.github.com/repos/df2test/my-test
-repo/actions/runs?branch=main\u0026per_page=1"},"timestamp":1748332331}
```

Standard Error

```
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loading config: /usr/local/packages/TelegrafAgent/localdata/configs/config_agent.conf
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loading config: /usr/local/packages/TelegrafAgent/localdata/configs/config_input_github.conf
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loading config: /usr/local/packages/TelegrafAgent/localdata/configs/config_output_stdout.conf
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Starting Telegraf 1.33.2 (customized) brought to you by InfluxData the makers of InfluxDB
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Available plugins: 18 inputs, 8 aggregators, 15 processors, 9 parsers, 8 outputs, 0 secret-stores
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loaded inputs: http
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loaded aggregators:
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loaded processors:
[2025-05-27 09:52:01] 2025-05-27T07:52:01Z I! Loaded secretstores:
```

Figure 3: UI logs page with stdout and stderr

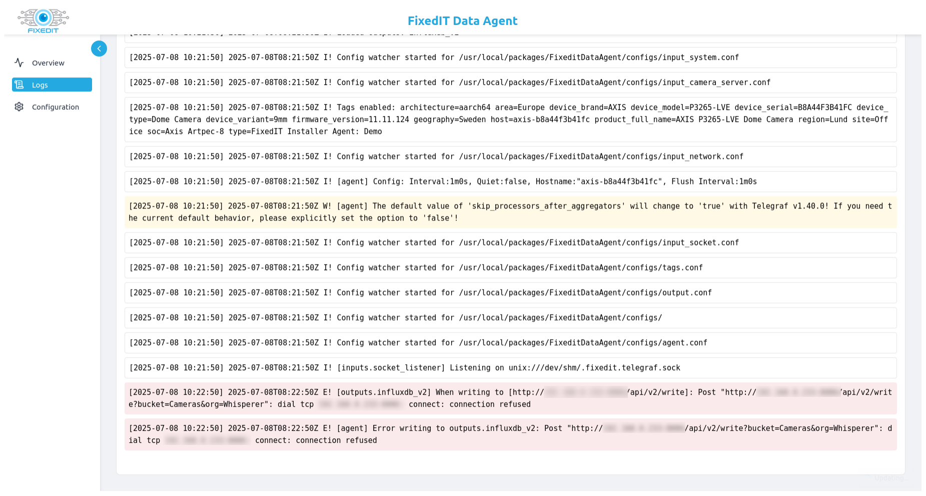
The default configuration used for Telegraf makes all the log messages related to the Telegraf process be logged to `stderr`, which leaves `stdout` available for adding custom logging output of metrics or other data which might be useful when using custom config files in the application.

Configuration page

The `Configuration` page can be used to manage the configuration files used by the Telegraf process. This includes viewing the content of the currently available configuration files in the camera, adding new files, removing files, or enabling/disabling files. A file being enabled means that it is used by the Telegraf process. If it is disabled, it means that it is stored in the camera, but not used for anything.

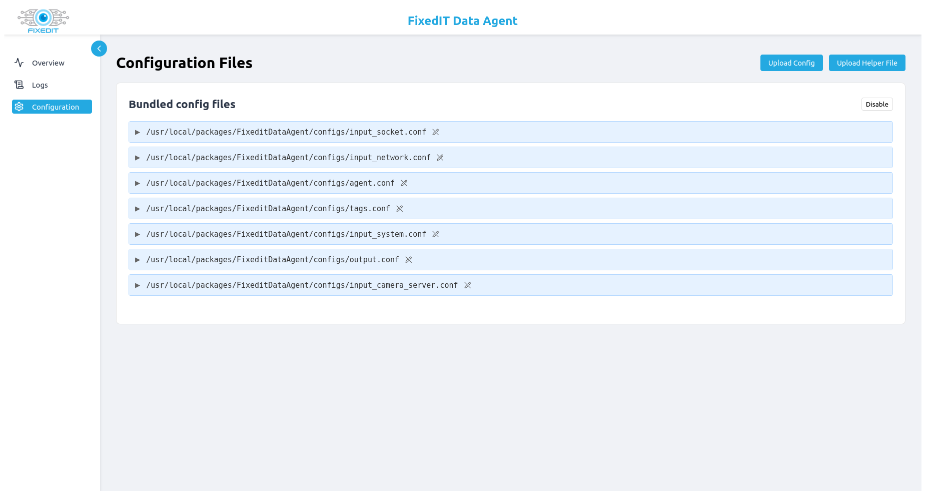
Bundled config files

Before any new files are uploaded, the page will only show the bundled config files in the UI. Files are listed using their full path. The icon of a crossed off



```
[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/input_system.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/input_camera_server.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Tags enabled: architecture=aarch64 area=Europe device_brand=AXIS device_model=P3265-LVE device_serial=B8A44F3B41FC device_
type=Dome Camera device_variant=9mm firmware_version=11.11.124 geography=Sweden host=axis-b8a44f3b41fc product_full_name=AXIS P3265-LVE Dome Camera region=Lund site=Off
ice soc=Axis Artpec-8 type=FixedIT Installer Agent: Demo

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/input_network.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! [agent] Config: Interval:1m0s, Quiet:false, Hostname:"axis-b8a44f3b41fc", Flush Interval:1m0s

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z W! [agent] The default value of 'skip_processors_after_aggregators' will change to 'true' with Telegraf v1.40.0! If you need t
he current default behavior, please explicitly set the option to 'false'!

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/input_socket.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/tags.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/output.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! Config watcher started for /usr/local/packages/FixeditDataAgent/configs/agent.conf

[2025-07-08 10:21:50] 2025-07-08T08:21:50Z I! [inputs.socket_listener] Listening on unix:///dev/shm/.fixedit.telegraf.sock

[2025-07-08 10:22:50] 2025-07-08T08:22:50Z E! [outputs.influxdb_v2] When writing to [http://[redacted]/api/v2/write]: Post "http://[redacted]/api/v2/writ
e?bucket=Cameras&org=Whisperer": dial tcp [redacted] connect: connection refused

[2025-07-08 10:22:50] 2025-07-08T08:22:50Z E! [agent] Error writing to outputs.influxdb_v2: Post "http://[redacted]/api/v2/write?bucket=Cameras&org=Whisperer": d
ial tcp [redacted] connect: connection refused
```

Figure 4: Example of errors and warnings in the logs page



Configuration Files

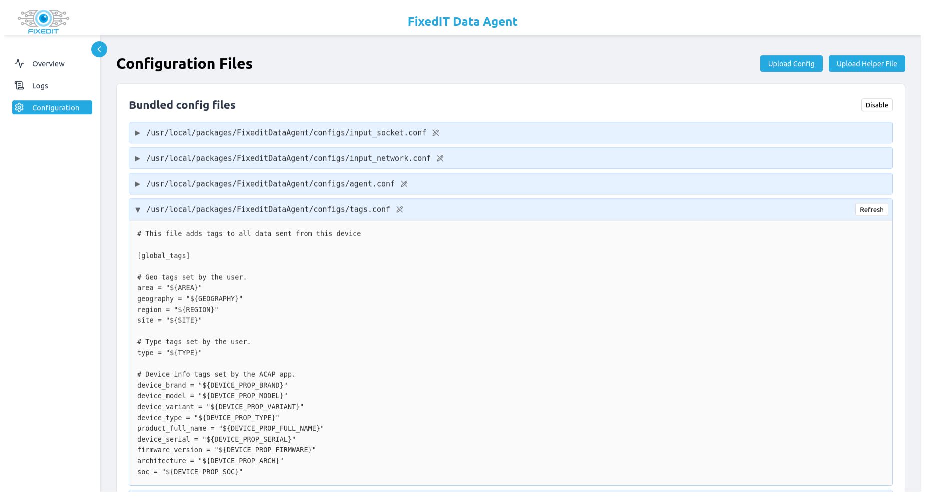
Bundled config files

```
▶ /usr/local/packages/FixeditDataAgent/configs/input_socket.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_network.conf
▶ /usr/local/packages/FixeditDataAgent/configs/agent.conf
▶ /usr/local/packages/FixeditDataAgent/configs/tags.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_system.conf
▶ /usr/local/packages/FixeditDataAgent/configs/output.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_camera_server.conf
```

Figure 5: Configuration page

pencil to the right of each file indicates that the file is read-only.

The content of the files can be viewed by clicking on the arrow to the left of each filename.



```
▶ /usr/local/packages/FixeditDataAgent/configs/input_socket.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_network.conf
▶ /usr/local/packages/FixeditDataAgent/configs/agent.conf
▼ /usr/local/packages/FixeditDataAgent/configs/tags.conf

# This file adds tags to all data sent from this device

[global_tags]

# Geo tags set by the user.
area = "${AREA}"
geography = "${GEOGRAPHY}"
region = "${REGION}"
site = "${SITE}"

# Type tags set by the user.
type = "${TYPE}"

# Device info tags set by the ACAP app.
device_brand = "${DEVICE_PROP_BRAND}"
device_model = "${DEVICE_PROP_MODEL}"
device_variant = "${DEVICE_PROP_VARIANT}"
device_type = "${DEVICE_PROP_TYPE}"
product_full_name = "${DEVICE_PROP_FULL_NAME}"
device_serial = "${DEVICE_PROP_SERIAL}"
firmware_version = "${DEVICE_PROP_FIRMWARE}"
architecture = "${DEVICE_PROP_ARCH}"
soc = "${DEVICE_PROP_SOC}"
```

Figure 6: Config file content window

When the content of a file is viewed, the `Refresh` button also appears to the right of the filename. Clicking on this will update the content of the file shown in the UI to the latest version of the file in the camera.

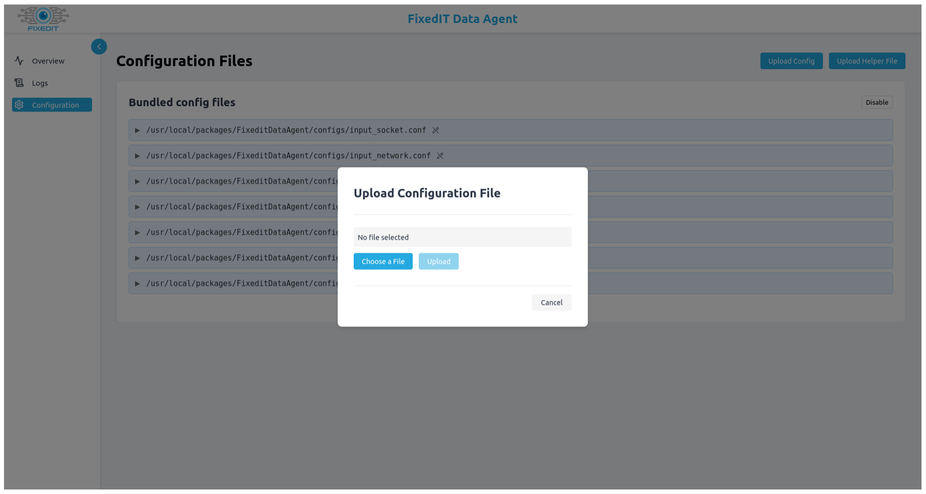
These files can be disabled and enabled as a group, i.e. either all of them or none of them are used by Telegraf. The bundled files are updated when the application is updated, and the content and dependencies of the config files might be refactored. Only having some of the bundled files enabled might break functionality after an update, so to ensure compatibility, they are treated as a group. They can be disabled/enabled using the `Disable`/`Enable` button at the top right of the `Bundled config files` section. If you only want to use some of the bundled files, it is recommended to disable all bundled files and upload your own copies of the files you want to use.

Uploading new config files

In addition to the bundled configuration files, you can upload new configuration files. This can be done by clicking the `Upload Config` button located in the top right corner of the UI. This will open up a pop-up window to choose a file to upload.

After uploading the file, it will show up under the `Uploaded config files` section.

The pencil icon to the right of each file indicates that the file is modifiable.



Configuration Files

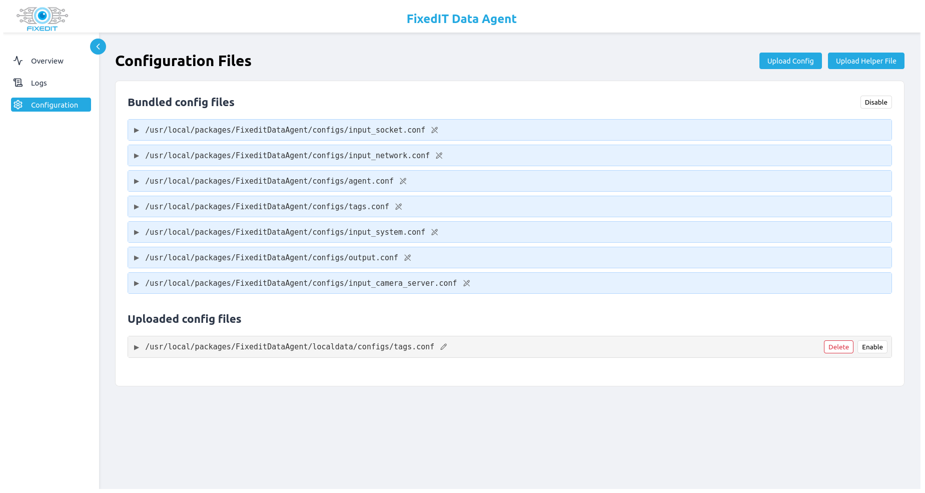
Bundled config files

```
▶ /usr/local/packages/FixeditDataAgent/configs/input_socket.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_network.conf
▶ /usr/local/packages/FixeditDataAgent/confi
▶ /usr/local/packages/FixeditDataAgent/confi
▶ /usr/local/packages/FixeditDataAgent/confi
▶ /usr/local/packages/FixeditDataAgent/confi
▶ /usr/local/packages/FixeditDataAgent/confi
```

Upload Configuration File

No file selected

Figure 7: Config file upload pop-up window



Configuration Files

Bundled config files

```
▶ /usr/local/packages/FixeditDataAgent/configs/input_socket.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_network.conf
▶ /usr/local/packages/FixeditDataAgent/configs/agent.conf
▶ /usr/local/packages/FixeditDataAgent/configs/tags.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_system.conf
▶ /usr/local/packages/FixeditDataAgent/configs/output.conf
▶ /usr/local/packages/FixeditDataAgent/configs/input_camera_server.conf
```

Uploaded config files

```
▶ /usr/local/packages/FixeditDataAgent/localdata/configs/tags.conf
```

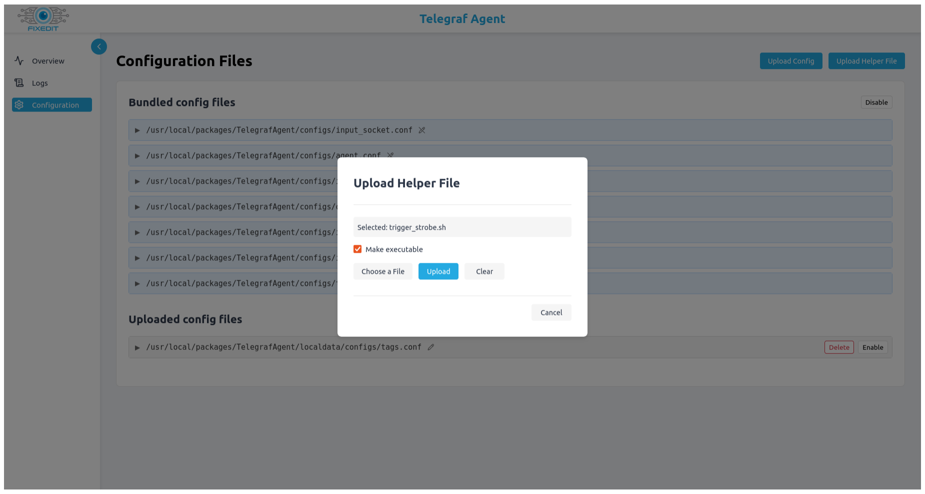
Figure 8: Uploaded config file

Each uploaded config file will show two buttons to the right: `Delete` and `Enable`.

- The `Delete` button removes the file from the camera completely, after removing the file it will also be removed from the list in the UI.
- The `Enable` button will make Telegraf use the config file. By default, Telegraf will not use the uploaded file until it is enabled. The file can be disabled again after enabling it. Files that are currently enabled cannot be deleted, they need to be disabled first.

Uploading new helper files

Some configurations might require additional files. These files can be uploaded by clicking on the `Upload Helper File` button located in the top right corner.



Configuration Files

Bundled config files

```
▶ /usr/local/packages/TelegrafAgent/configs/input_socket.conf
▶ /usr/local/packages/TelegrafAgent/configs/agent.conf
▶ /usr/local/packages/TelegrafAgent/configs/
▶ /usr/local/packages/TelegrafAgent/configs/
▶ /usr/local/packages/TelegrafAgent/configs/
▶ /usr/local/packages/TelegrafAgent/configs/
▶ /usr/local/packages/TelegrafAgent/configs/
```

Uploaded config files

```
▶ /usr/local/packages/TelegrafAgent/localdata/configs/tags.conf
```

Upload Helper File

Selected: trigger_strobe.sh

Figure 9: Helper file upload pop-up window

The file can be made executable by checking the `Make executable` option. Once uploaded, the file will show up under `Uploaded helper files`. Executable files will be shown in green, as shown below.

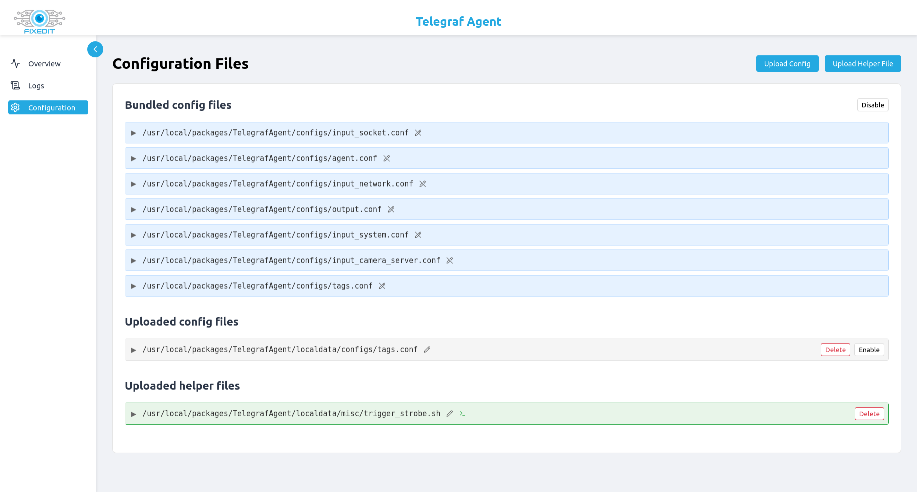
The pencil icon to the right of each file indicates that the file is modifiable. The code icon indicates that the file is executable.

Like for config files, a `Delete` button will appear to the right of each uploaded helper file. Clicking on this removes the file from the camera and UI.

Helper files are never used directly by the application, only by the config files. Therefore, they do not have `Enable`/`Disable` buttons.

Example use of config files with helper files

Let's take the following config file as an example.



Telegraf Agent

Configuration Files

Bundled config files

```
▶ /usr/local/packages/TelegrafAgent/configs/input_socket.conf
▶ /usr/local/packages/TelegrafAgent/configs/agent.conf
▶ /usr/local/packages/TelegrafAgent/configs/input_network.conf
▶ /usr/local/packages/TelegrafAgent/configs/output.conf
▶ /usr/local/packages/TelegrafAgent/configs/input_system.conf
▶ /usr/local/packages/TelegrafAgent/configs/input_camera_server.conf
▶ /usr/local/packages/TelegrafAgent/configs/tags.conf
```

Uploaded config files

```
▶ /usr/local/packages/TelegrafAgent/localdata/configs/tags.conf
```

Uploaded helper files

```
▶ /usr/local/packages/TelegrafAgent/localdata/misc/trigger_strobe.sh
```

Figure 10: Uploaded helper file

```
# Configuration for executing a script
[[outputs.exec]]
  # Command to execute when metrics are received
  command = ["${HELPER_FILES_DIR}/trigger_strobe.sh"]

  # Data format to use for input
  data_format = "json"
```

It runs the `trigger_strobe.sh` script when metrics are received. The script needs to be uploaded as a helper file. All helper files are uploaded to the same directory in the camera, and the application is setting the environment variable `HELPER_FILES_DIR` to this path. This variable can then be used from configuration files as seen in the example above, in the following line:

```
  command = ["${HELPER_FILES_DIR}/trigger_strobe.sh"]
```

For a more complex use case example involving config files and helper files, see this video.

Application configuration

This section describes different parameters that can be used to configure how the application works. These can e.g. be changed from the camera's UI, in the application's `Settings` tab.

These parameters can also be configured in bulk through automation (for example, using VAPIX).

Note that some of these parameters are considered part of the advanced appli-

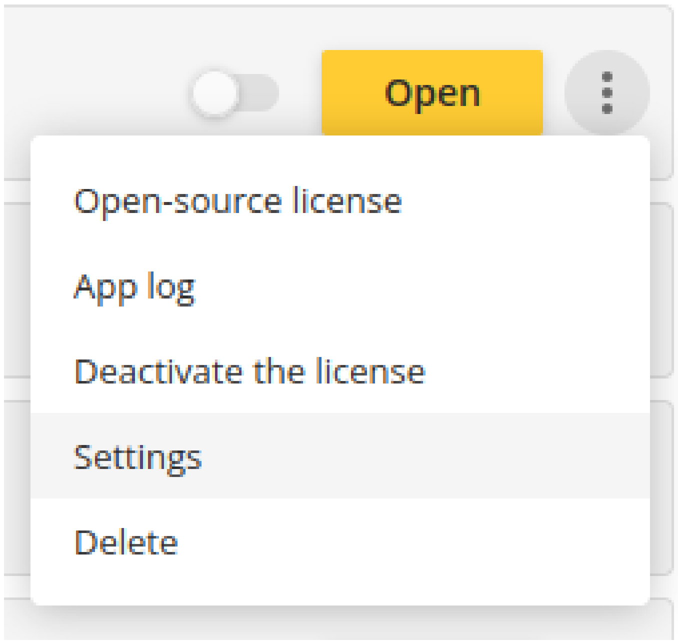


Figure 11: Settings tab

cation configuration and are explained in the section below.

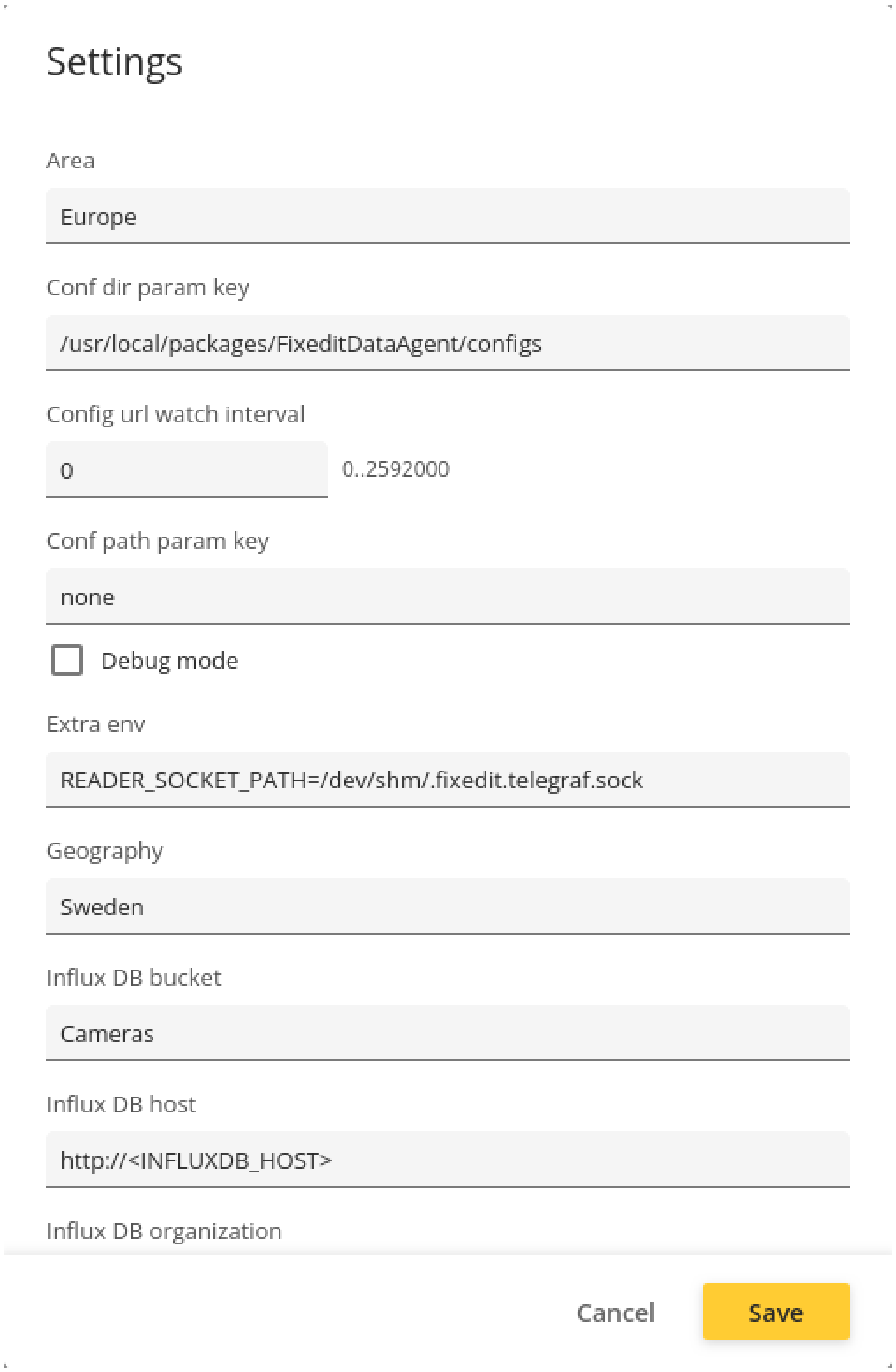
Debug mode configuration

- `DebugMode`: Enables or disables debug logging in Telegraf. When enabled, Telegraf will output more verbose logging information, which can be useful for troubleshooting configuration issues. This will set the `TELEGRAF_DEBUG` environment variable.

InfluxDB instance configuration

Since a common use case is to send data to an InfluxDB instance, the following parameters are available for simplicity and will be used by default by the `output.conf` file.

- `InfluxDBHost`: Hostname or IP address of the InfluxDB instance including the protocol (e.g. `http://192.168.0.123` or `https://influxdb.example.com`), this will set the `INFLUX_HOST` environment variable.
- `InfluxDBPort`: Port number of the InfluxDB instance, this will set the `INFLUX_PORT` environment variable.
- `InfluxDBToken`: Authentication token for accessing InfluxDB, this will set the `INFLUX_TOKEN` environment variable.



Settings

Area

Europe

Conf dir param key

/usr/local/packages/FixeditDataAgent/configs

Config url watch interval

0 0..2592000

Conf path param key

none

☐ Debug mode

Extra env

READER_SOCKET_PATH=/dev/shm/.fixedit.telegraf.sock

Geography

Sweden

Influx DB bucket

Cameras

Influx DB host

http://<INFLUXDB_HOST>

Influx DB organization

Figure 12: Settings window

- `InfluxDBOrganization`: Name of the InfluxDB organization, this will set the `INFLUX_ORG` environment variable.
- `InfluxDBBucket`: Name of the InfluxDB bucket where metrics are stored, this will set the `INFLUX_BUCKET` environment variable.

Since the Telegraf process can make use of environment variables, these can then be used e.g. like this in the config files:

```
[[inputs.cpu]]
  percpu = true
  totalcpu = true
  name_override = "${KEY2}"

[[outputs.influxdb_v2]]
  urls = ["${INFLUX_HOST}:${INFLUX_PORT}"]
  token = "${INFLUX_TOKEN}"
  organization = "${INFLUX_ORG}"
  bucket = "${INFLUX_BUCKET}"
```

Message tag configuration

To easier keep track of your cameras, we recommend using geo tags and type tags. These will by default be used by the `tags.conf` file. Despite our recommendations on how to use the geo tags, there is no enforcement and you are free to group the geography in whatever way makes sense to your business.

- `Area`: The top-most geo, e.g. `Europe`
- `Geography`: The second level, e.g. `Sweden`
- `Region`: The third level, e.g. `Stockholm`
- `Site`: The specific location, e.g. `Arena 5`
- `Type`: The type of installation, e.g. `Surveillance` or `Analytics`

Advanced application configuration

The following are parameters available in the application settings tab which relate to more advanced configuration.

Parameters for setting Telegraf configuration files

These parameters can be used to specify which configuration files Telegraf should use. All configuration files will be used in the same process, so note that they will essentially be merged by Telegraf into one single configuration file affecting the same process. This is the same as running Telegraf with multiple configuration files, e.g. `telegraf --config config1.conf --config config2.conf`.

- `ConfPathParamKey`: Absolute paths to the Telegraf configuration files, separated by ";".

- `ConfDirParamKey`: Absolute path to a folder containing configuration files. All configuration files in the folder will be used by Telegraf.

It is not recommended to manually configure these settings if the UI is also being used to upload and configure files, as they might interfere. See the note about uploading configuration files in the Telegraf config files section.

Currently, there is no officially supported way to upload custom configuration files in bulk or using automation.

Setting custom environment variables

Environment variables can be configured in the application and will be readable from the Telegraf configuration files. Some environment variables used frequently have their own parameters for simplicity, as seen in the Application configuration section, but more can be specified using the `ExtraEnv` parameter.

- `ExtraEnv`: User defined variables, key-value pairs separated by ";". E.g.: `KEY=value;KEY2=value2`. These will be set in Telegraf's environment and can be used from the configuration files.

In the Application configuration section, it is explained that setting some of the parameters also sets a corresponding environment variable (e.g. setting `InfluxDBHost` will set the `INFLUX_HOST` environment variable). Note that those environment variables must not be specified in the `ExtraEnv` parameter.

Automatically set environment variables

The application automatically sets several environment variables for the Telegraf subprocess that are not configurable by users:

System environment variables:

- `HELPER_FILES_DIR`: Set to the directory where helper files are stored, useful for referencing helper files in Telegraf configuration files.
- `HOME`: Set to a dummy directory to avoid Telegraf error messages about missing home directory. This should not be used by the user.

Device information variables: The following environment variables are automatically set with device-specific information read from the camera's parameter system:

- `DEVICE_PROP_BRAND`: Device brand (e.g., "AXIS")
- `DEVICE_PROP_MODEL`: Device model number (e.g., "M1075-L")
- `DEVICE_PROP_VARIANT`: Device variant (in most cases, this will be empty)
- `DEVICE_PROP_TYPE`: Device type (e.g., "Box Camera")
- `DEVICE_PROP_FULL_NAME`: Full product name (e.g., "AXIS M1075-L Box Camera")
- `DEVICE_PROP_SERIAL`: Device serial number (e.g., "B8A44F717321")
- `DEVICE_PROP_FIRMWARE`: Firmware version (e.g., "12.5.56")

- `DEVICE_PROP_ARCH`: System architecture (e.g., "aarch64")
- `DEVICE_PROP_SOC`: System-on-chip information (e.g., "Ambarella CV25")

Telegraf config files

Telegraf config files are used to create tasks / services with the FixedIT Data Agent. A Telegraf config file contains at least one input and one output. It might also contain processors, parsers, aggregators and serializers. Telegraf can use multiple config files at the same time, but they will essentially be merged by Telegraf into one single configuration file affecting the same process. Using multiple config files is therefore useful to split different tasks to make it easier to manage them, but will not create independent tasks / services. A config file might e.g. say "Collect system metrics from the system (CPU usage, memory usage, etc.) and send them to InfluxDB."

While the application already contains some configuration files, new ones can also be copied to the camera in order for the application to use them. If your application supports the web UI, then you can upload, manage and delete config files from there. If your application does not, it can be configured from e.g. the FixedIT Installer Agent.

The application includes several bundled configuration files that provide comprehensive monitoring capabilities:

Bundled Configuration Files:

- `agent.conf`: Core Telegraf configuration with collection intervals and buffer settings
- `input_system.conf`: Collects system metrics (CPU, memory, disk, system uptime)
- `input_network.conf`: Monitors network connectivity, DNS, ping, and IP addresses
- `input_camera_server.conf`: Health checks for the camera's HTTP/HTTPS API endpoints
- `output.conf`: Sends metrics to InfluxDB using environment variables
- `tags.conf`: Adds geographic and device information tags to all metrics

When the application is installed in the device, these files are installed to the `/usr/local/packages/FixeditDataAgent/configs/` directory and are used by default as long as no custom configuration has changed it.

To specify which config files to use, the `ConfPathParamKey` and `ConfDirParamKey` parameters can be set to the paths in the camera that they have been copied to. By default, the `ConfPathParamKey` will point to the configuration files included in the application. You can use both of these simultaneously.

Creating and testing custom configuration files on host

If you want to develop and test a telegraf config file, then the easiest way is to install Telegraf locally and run your config directly on your host machine. E.g. on Linux you can run:

```
sudo apt-get install telegraf
telegraf --config <my-config-file.conf>
```

Config files may make use of environment variables (e.g. `${INFLUX_HOST}`) in the config file. If so, you need to export these variables before running the config file.

```
export INFLUX_HOST=http://localhost
export INFLUX_PORT=8086
export INFLUX_TOKEN=<my-token>
export INFLUX_ORG=test
export INFLUX_BUCKET=test_bucket
telegraf --config app/configs/input_system.conf
```

Debugging configuration files

One way to make debugging easier is to add the following to the config file:

```
[[outputs.file]]
  files = ["stdout"]
  data_format = "influx"
```

This will cause Telegraf to write all captured data to stdout. The application will log all standard output and errors to the application's UI, so this can be useful when debugging how a configuration file works with the application as well.

Telegraf plugin overview

Telegraf used in the agent has been compiled with a specific set of plugins, these are the only ones that you can use in your config files when running them in the FixedIT Data Agent. The following is the complete list of plugins available in this release.

Aggregators (statistical data processing)

Aggregators process multiple data points over a time window before sending them to outputs. They reduce noise and provide useful summaries of raw data.

Plugin	Description	When to Use It?
basicstats	Computes mean, min, max, standard deviation, etc.	Useful for summarizing sensor readings before sending data upstream.
derivative	Computes the rate of change between data points.	Ideal for monitoring network throughput, power usage, or sensor variations.
final	Outputs only the last value in an aggregation window.	Reduces unnecessary data transmission in low-bandwidth IoT networks.
histogram	Generates bucketed distribution data.	Useful for latency analysis in network monitoring.
merge	Merges multiple metrics into one.	Helpful for combining sensor readings from different data sources.
minmax	Tracks min/max values within a time window.	Useful for detecting spikes and anomalies.
quantile	Computes percentile-based statistics.	Great for latency and performance monitoring.
valuecounter	Counts occurrences of unique values.	Detects event frequency, such as error counts or device state changes.

Inputs (data collection)

These plugins collect data from various sources, including system metrics, network devices, and logs.

Plugin	Description	When to Use It?
cpu, mem, disk, diskio, swap, system	Collects CPU, memory, disk usage, I/O, and system uptime.	Essential for monitoring the health of edge devices.
net, netstat	Monitors network bandwidth, TCP connections, and packet statistics.	Helps detect connectivity issues or abnormal traffic.
file	Reads and processes structured log files.	Useful for log-based monitoring.
tail	Reads and processes log files as they are written to.	Useful for log-based monitoring.
http_listener_v2	Collects metrics via HTTP.	Ideal for receiving metrics from external sources.
http_response	Measures HTTP response time and status codes.	Helps monitor API health.

Plugin	Description	When to Use It?
http	Make a HTTP request to get the content of a web page.	Good for fetching data from web APIs.
influxdb	Collects data from InfluxDB databases.	Useful for time-series analytics.
processes	Monitors running processes and resource usage.	Helps detect abnormal resource consumption.
socket__listener	Receives metrics via a socket.	Enables custom integrations.
exec, execd	Runs a command and captures the output.	Useful for running custom commands when there is no plugin to do what you want to do.
mqtt__consumer	Subscribes to a MQTT broker and consumes messages.	Useful for receiving messages from a server or other applications and acting on them.

Outputs (data forwarding)

These plugins send collected data to various destinations, including databases, logs, and cloud platforms.

Plugin	Description	When to Use It?
influxdb, influxdb__v2	Stores time-series data.	Best for long-term storage and analytics.
mqtt	Publishes metrics to MQTT brokers.	Ideal for IoT message streaming.
exec	Sends data to external scripts for processing.	Enables custom data handling.
file	Saves data to a local file.	Useful for local debugging and backups.
remotefile	Saves metrics to a remote file.	Helps in distributed logging.
syslog	Sends logs to a syslog server.	Useful for log aggregation.
websocket	Streams data via WebSockets.	Enables real-time dashboards.

Processors (data transformation)

Processors modify, enrich, or filter incoming data before sending it to outputs.

Plugin	Description	When to Use It?
converter, scale, override	Converts values, scales numbers, and overrides fields.	Useful for standardizing sensor data.
rename, dedup, tag_limit	Renames fields, removes duplicates, and limits tags.	Reduces data noise and unnecessary processing.
filter, lookup, defaults	Filters out unwanted data and applies default values.	Helps in reducing bandwidth usage.
reverse_dns	Resolves hostnames from IP addresses.	Required for networked device monitoring.
template, timestamp, date	Modifies timestamps and restructures metrics.	Useful for time-series data formatting.
starlark, execd	Allows custom scripting and external processing.	Ideal for complex IoT data manipulation.

Parsers (data format handling)

Parsers convert data from raw formats into structured Telegraf metrics.

Plugin	Description	When to Use It?
binary	Parses binary data into structured metrics.	Useful for decoding binary data.
influx	Parses InfluxDB line protocol.	If importing metrics from InfluxDB sources.
json	Parses JSON data.	For parsing simple json data.
json_v2	Parses JSON data with advanced structuring.	Best for parsing complex json data.
value	Parses simple data types like string and int.	For parsing when the whole input is a single type
grok	Parses log files using custom patterns.	For parsing log files.

Serializers (data format for output)

Serializers format data before sending it to outputs.

Plugin	Description	When to Use It?
influx	Formats data in InfluxDB line protocol.	For compatibility with InfluxDB.
json	Formats data in JSON format.	For cloud and API integrations.
csv	Formats data in CSV format.	For spreadsheet and data analysis.

Plugin	Description	When to Use It?
binary	Formats data in binary format.	For performance-critical applications.

Changelog

1.0.1

- Bump to manifest schema version 1.3 (AXIS OS 10.9+).

1.0.0

- Add `mqtt_consumer` input plugin to the application.
- Add `execd` input plugin to the application.
- Add the `HELPER_FILES_DIR` environment variable to the application.
- Complete the rebranding effort to "FixedIT Data Agent".
- Remove the bundled socket listener configuration file.

0.0.4

- Add configuration parameter for enabling debug mode.
- Add better logging to the system log.
- Correct license notices for third-party software.
- Fix bug to enable trailing slashes in INFLUXDB_HOST parameter.

0.0.3

- Fix bugs in bundled telegraf config files
- Delay telegraf restart when settings are changed to avoid multiple restarts
- Run FixedIT Data Agent as a dynamic user for increased security and compatibility
- Add socket input to read messages from other ACAP applications

0.0.2

- Set default InfluxDB port to 8086
- Include protocol in InfluxDB host parameter (e.g. `http://192.168.0.123` instead of `192.168.0.123`) - this makes it possible to use both HTTP for self-hosted dev servers and HTTPS for servers in production.

0.0.1

- Initial release with basic functionality and simple web UI.