

File system log appender

- [Creating file system log appender in Admin UI](#)
- [Creating file system log appender with REST API](#)
 - [Configuration](#)
- [Administration](#)
- [Using with File system log appender](#)

The file system log appender stores received logs into the local file system of the Operations server. This log appender may be used for test purposes or in pair with tools like Flume and others. Logs are stored in files under the `/${logsRootPath}/tenant_${tenantId}/application_${applicationId}` folder, where `logsRootPath` is a configuration parameter, `tenantId` and `applicationId` are ids of the current tenant and the application respectively. Access to the logs is controlled via Linux file system permissions.

You can log in to the Operations server host and browse logs using the `kaa_log_user_${applicationToken}` user name and the public key which is created as a part of the configuration.

Creating file system log appender in Admin UI

The easiest way to create a file system log appender for your application is by using [Admin UI](#).

Creating file system log appender with REST API

It is also possible to create a file system log appender for your application by using REST API. The following example illustrates how to provision the file system log appender via REST API.

Configuration

The file system log appender configuration should match the following Avro schema.

▼ [Click here to expand...](#)

```

{
  "namespace": "org.kaaproject.kaa.server.appenders.file.config.gen",
  "type": "record",
  "name": "FileConfig",
  "fields": [
    {
      "name": "publicKey",
      "displayName": "Public Key",
      "maxLength": 1000,
      "default": "",
      "type": "string"
    },
    {
      "name": "logsRootPath",
      "displayName": "Logs root path",
      "default": "/kaa_log_uploads",
      "type": "string"
    },
    {
      "name": "rollingFileNamePatern",
      "displayName": "Rolling file name pattern",
      "default": "logFile.%d{yyyy-MM-dd}.log",
      "type": "string"
    },
    {
      "name": "rollingMaxHistory",
      "displayName": "Rolling max history",
      "default": 30,
      "type": "int"
    },
    {
      "name": "triggerMaxFileSize",
      "displayName": "Trigger max file size",
      "default": "1GB",
      "type": "string"
    },
    {
      "name": "encoderPattern",
      "displayName": "Encoder pattern",
      "default": "%-4relative [%thread] %-5level %logger{35} - %msg%n",
      "type": "string"
    }
  ]
}

```

name	description
publicKey	Name of public key
logsRootPath	Root path for logs
rollingFileNamePatern	Pattern for creating file name
rollingMaxHistory	Max number for records in file

triggerMaxFileSize	Max size of file
encoderPattern	Pattern for encoder

The following configuration example matches the previous schema.

▼ [Click here to expand...](#)

```
{
  "publicKey": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "logsRootPath": "/kaa_log_uploads",
  "rollingFileNamePattern": "logfile.%d{yyyy-MM-dd}.log",
  "rollingMaxHistory": 30,
  "triggerMaxFileSize": "1GB",
  "encoderPattern": "%-4relative [%thread] %-5level %logger{35} - %msg%n"
}
```

Administration

The following REST API call example illustrates how to create a new file system log appender.

```
curl -v -S -u devuser:devuser123 -X POST -H 'Content-Type: application/json'
-d'{"pluginClassName":
"org.kaaproject.kaa.server.appenders.file.appender.FileSystemLogAppender",
"applicationId": 119, "applicationToken": "91786338058670361194",
"jsonConfiguration":
"{\"publicKey\": \"XXXXXXXXXXXXXXXXXXXXXXXXXXXX\", \"logsRootPath\": \"/kaa_log_uploads\",
\", \"rollingFileNamePattern\": \"logfile.%d{yyyy-MM-dd}.log\", \"rollingMaxHistory\": 30,
\"triggerMaxFileSize\": \"1GB\", \"encoderPattern\": \"%-4relative [%thread] %-5level
%logger{35} - %msg%n\" }", "description": "New sample file system log appender",
"headerStructure": [ "KEYHASH", "TIMESTAMP" ], "name": "New file system appender",
"maxLogSchemaVersion": 2147483647, "minLogSchemaVersion": 1, "tenantId": "70"}'
"http://localhost:8080/kaaAdmin/rest/api/logAppender" | python -mjson.tool
```

▼ [Example result](#)

```

{
  "appenderClassName":
  "org.kaaproject.kaa.server.appenders.file.appender.FileSystemLogAppender",
  "applicationId": "70",
  "applicationToken": "946558468095768",
  "configuration":
  "{\\"publicKey\\":\\"XXXXXXXXXXXXXXXXXXXXXXXXXXXX\\",\\"logsRootPath\\":\\"/kaa_log_upload
s_new\\",\\"rollingFileNamePatern\\":\\"logFile.%d{yyyy-MM-dd}.log\\",\\"rollingMaxHist
ory\\":30,\\"triggerMaxFileSize\\":\\"1GB\\",\\"encoderPattern\\":\\"%-4relative
[%thread] %-5level %logger{35} - %msg%n\\"}",
  "createdTime": 1417006362287,
  "createdUsername": "devuser",
  "description": "New sample file system log appender",
  "headerStructure": [
    "KEYHASH",
    "TIMESTAMP"
  ],
  "id": "161",
  "name": "New file system appender",
  "maxLogSchemaVersion": 2147483647,
  "minLogSchemaVersion": 1,
  "status": "REGISTERED",
  "tenantId": "10",
  "typeName": "File"
}

```

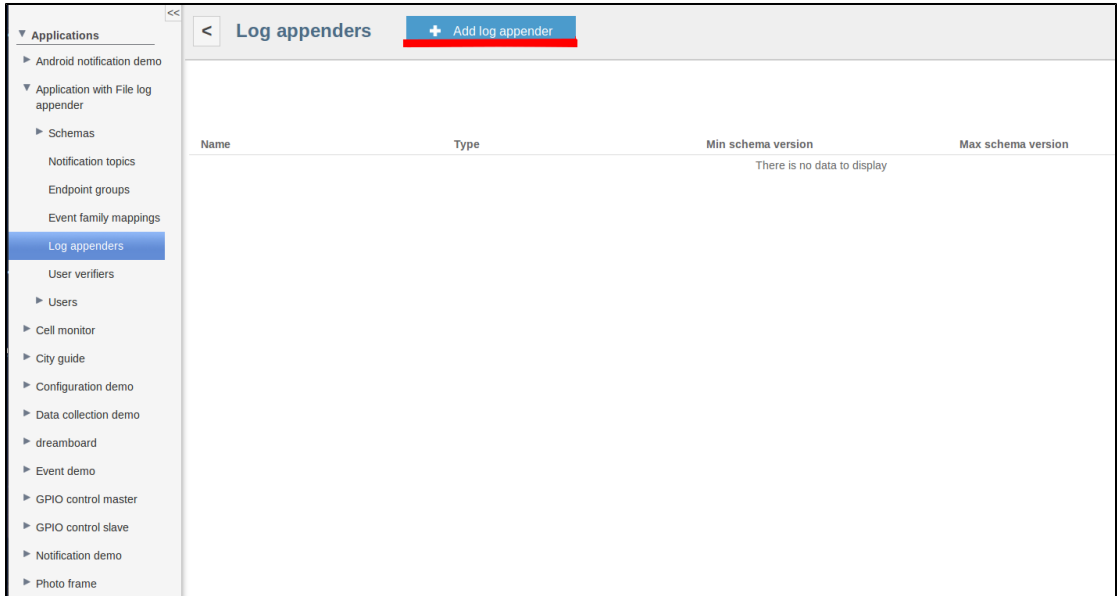
Using with File system log appender

You can create your application where you will use file system log appender

For this you should create application in Admin UI

The screenshot shows the Kaa Admin Console interface. On the left is a sidebar with a navigation menu containing 'Applications', 'Users', and 'Event class families'. The main area is titled 'Admin Console' and features a 'Add application' button with 'Add' and 'Cancel' sub-buttons. Below this is the 'Application details' section, which includes a note: 'Fields marked with * are mandatory.' The 'Title *' field is filled with the text 'Application with File log appender' and has a character count of '34 of 255 max characters' displayed below it.

After that add new appender to your application



Enter name of the new appender

Select **File** appender type.



Add log appender

Add

Cancel

Log appender details

Fields marked with * are mandatory.

Name *

My log appender

15 of 255 max characters

Min schema version *

1

Max schema version *

Infinite

Confirm delivery



Log metadata

Select metadata components

Description

0 of 1024 max characters

Type

File

Add new parameters of configuration or replace old.

Public Key *	<input type="text" value="Some Public Key"/>
	15 of 1000 max characters
Logs root path *	<input type="text" value="/kaa_log_uploads"/>
Rolling file name pattern *	<input type="text" value="logfile.%d{yyyy-MM-dd}.log"/>
Rolling max history *	<input type="text" value="30"/>
Trigger max file size *	<input type="text" value="1GB"/>
Encoder pattern *	<input type="text" value="%-4relative [%thread] %-5level %logger{35} - %msg"/>

Now click **Add** button on the top of the screen to create and deploy appender.

Add log appender

Verify that newly created appender has appeared in list.

Log appenders

Name	Type	Min schema version	Max schema version
My log appender	File	1	Infinite

After that you can go to Data collection demos in Sandbox.

Sandbox

Data collection Java demo

Kaa Logging subsystem demo app based on Java SDK

SDK language Java

Platforms Linux x86 Windows x86

Features Data collection

Complexity Basic

[Source](#)
[Binary](#)

This simple app demonstrates capabilities of the Kaa Logging subsystem on the Java SDK.

Installation

Download the jar file to your device by clicking the "Binary" button on the left. Make sure that you have java installed on your device. Run the application using the following command in the console:

```
$ java -jar DataCollectionDemo.jar
```

Run the application using the following command in the console:

```
$ java -jar DataCollectionDemo.jar
```

```
chvova@chvova:~/Downloads$ java -jar DataCollectionDemo.jar
2016-02-28 16:51:26,573 [main] INFO o.k.k.d.d.DataCollectionDemo - Data collection de
2016-02-28 16:51:27,628 [pool-2-thread-1] INFO o.k.k.d.d.DataCollectionDemo - Kaa cli
2016-02-28 16:51:27,633 [main] INFO o.k.k.d.d.DataCollectionDemo - Log record {"level
2016-02-28 16:51:27,634 [main] INFO o.k.k.d.d.DataCollectionDemo - Log record {"level
2016-02-28 16:51:27,634 [main] INFO o.k.k.d.d.DataCollectionDemo - Log record {"level
2016-02-28 16:51:27,635 [main] INFO o.k.k.d.d.DataCollectionDemo - Log record {"level
2016-02-28 16:51:27,635 [main] INFO o.k.k.d.d.DataCollectionDemo - Log record {"level
2016-02-28 16:51:29,191 [main] INFO o.k.k.d.d.DataCollectionDemo - Received log recor
2016-02-28 16:51:29,191 [main] INFO o.k.k.d.d.DataCollectionDemo - Received log recor
2016-02-28 16:51:29,191 [main] INFO o.k.k.d.d.DataCollectionDemo - Received log recor
2016-02-28 16:51:29,191 [main] INFO o.k.k.d.d.DataCollectionDemo - Received log recor
2016-02-28 16:51:29,192 [main] INFO o.k.k.d.d.DataCollectionDemo - Received log recor
2016-02-28 16:51:29,193 [pool-2-thread-1] INFO o.k.k.d.d.DataCollectionDemo - Kaa cli
2016-02-28 16:51:29,197 [main] INFO o.k.k.d.d.DataCollectionDemo - Data collection de
```

This logs you can find in `/kaa_log_uploads/tenant_'number of tenant'/application_'your_application_token'/application.log`

In this example `/kaa_log_uploads/tenant_70/application_46527342666485986401/application.log`

```
5172492 [EPS-log-dispatcher-10] INFO 70.46527342666485986401 -
{"Log Header": {"endpointKeyHash":null,"applicationToken":null,"headerVersion":null,"t
"Event": {"level":"KAA_INFO","tag":"TAG","message":"MESSAGE_0"}}
5172502 [EPS-log-dispatcher-10] INFO 70.46527342666485986401 -
{"Log Header": {"endpointKeyHash":null,"applicationToken":null,"headerVersion":null,"t
"Event": {"level":"KAA_INFO","tag":"TAG","message":"MESSAGE_1"}}
5172502 [EPS-log-dispatcher-10] INFO 70.46527342666485986401 -
{"Log Header": {"endpointKeyHash":null,"applicationToken":null,"headerVersion":null,"t
"Event": {"level":"KAA_INFO","tag":"TAG","message":"MESSAGE_2"}}
5172502 [EPS-log-dispatcher-10] INFO 70.46527342666485986401 -
{"Log Header": {"endpointKeyHash":null,"applicationToken":null,"headerVersion":null,"t
"Event": {"level":"KAA_INFO","tag":"TAG","message":"MESSAGE_3"}}
5172502 [EPS-log-dispatcher-10] INFO 70.46527342666485986401 -
{"Log Header": {"endpointKeyHash":null,"applicationToken":null,"headerVersion":null,"t
"Event": {"level":"KAA_INFO","tag":"TAG","message":"MESSAGE_4"}}
```