

Encrypting Sensitive Properties

Date published: 2019-04-15

Date modified: 2022-04-27



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

| | |
|---|----------|
| Encrypt sensitive properties..... | 4 |
| Encryption of a single property..... | 4 |
| Encryption of multiple properties..... | 4 |
| Startup of EFM with encrypted properties..... | 5 |

Encrypt sensitive properties

You can encrypt sensitive information stored in the `efm.properties` file. You can encrypt a single sensitive property stored in the `efm.properties` file. You can also encrypt multiple sensitive properties stored in the `efm.properties` file.

Whenever you store sensitive information like database, keystore passwords in the `efm.properties` file, you are essentially making them vulnerable. Edge Flow Manager (EFM) provides an internal way to deal with this problem.

To decrypt sensitive properties EFM relies on the Jasypt library which Cloudera also recommends to use for encryption.

Encryption of a single property

Learn how you can encrypt a single sensitive property stored in the `efm.properties` file.

To encrypt a property, you should use the Jasypt CLI which you can download from here: <http://www.jasypt.org/cli.html>

Once you download the CLI, a single property can be encrypted by using the following command:

```
./encrypt.sh input=propertyValueToEncrypt password=secretKey \  
algorithm=PBEWITHHMACSHA512ANDAES_256 \  
ivGeneratorClassName=org.jasypt.iv.RandomIvGenerator
```

Please note that you need to specify the above algorithm and the `ivGeneratorClassName` values, because these are the default ones in Edge Flow Manager (EFM). If you do not specify them, the Jasypt CLI uses its defaults which is incompatible with EFM. If you still want to use them or if you want to choose a different algorithm, you need to specify them for EFM with these properties in the `efm.properties` file:

```
efm.property.encryptor.algorithm=  
efm.property.encryptor.ivGeneratorClassName=
```

To list available algorithms you can run the following command:

```
./listAlgorithms.sh
```

Please note that this command does not list any argument from non-default JCE providers like Bouncy Castle unless you have registered such providers at the JVM. For more info visit <http://www.jasypt.org/non-default-providers.html>.

You can use this approach to encrypt multiple properties, but you should use the same password for each as you can provide only one password when starting up EFM.

With some Java 8 versions, the Jasypt CLI throws `java.lang.ExceptionInInitializerError` error. To fix this issue you need to update the `icu4j` library used by Jasypt. This means you need to update the `<jasypt-root>/lib/icu4j-3.4.4.jar` file (if you are using Jasypt 1.9.3) with a newer version that you can download from here: https://github.com/unicode-org/icu/releases/download/release-68-2/icu4j-68_2.jar.

Once you acquire the encrypted value, you need to wrap it in `ENC(...)` to let EFM know that it should treat the property as an encrypted value. For example,

```
efm.server.ssl.keyStorePassword=ENC(e2cpfr5CA+xyS8uU2BNX1tKoR/hCBJeJlBxMAO21  
Ngt1snFOGza6uUCJCZGGN15Q)
```

Encryption of multiple properties

Learn how you can encrypt multiple sensitive properties stored in the `efm.properties` file.

Although it is possible to encrypt multiple properties one by one described in *Encryption of a single property* section, Edge Flow Manager (EFM) provides a helper script for convenience that you can use to encrypt all sensitive properties stored in the `efm.properties` file.

This script relies on the Jasypt CLI, therefore it is necessary that you install it. Also it requires the `JASYPT_HOME` environment variable set, which points to the root directory of Jasypt.

To encrypt a specific property, wrap its unencrypted value with `DEC(...)` in the `efm.properties` file. For example,

```
efm.server.ssl.keyStorePassword=DEC(passwordToEncrypt)
```

To encrypt the `efm.properties` file, you need to use the `encrypt_properties.sh` command and you need to provide the secret key with the `-p` flag which stands for password. For example,

```
./bin/encrypt_properties.sh -p secretKey
```

Unless it is explicitly specified through `-o` option, this script produces a `.encrypted` file next to the original one. You need to overwrite the original file with this newly created file.

You can also use the following optional flags:

- `-h`: Prints help
- `-a`: Specifies the algorithm used for encryption. Default value is `PBEWITHHMACSHA512ANDAES_256`.
- `-i`: Specifies the `ivGeneratorClassName`. Default value is `org.jasypt.iv.RandomIvGenerator`.
- `-l`: Specifies an alternative location for the property file. Default value is `conf/efm.properties`.
- `-o`: Specifies the output file location. Default value is `conf/efm.properties.encrypted (<originalfile>.encrypted)`.

Startup of EFM with encrypted properties

Learn how to start up Edge Flow Manager (EFM) after you encrypt a single sensitive property or multiple sensitive properties.

After you set the encrypted properties in the `efm.properties` file described in *Encryption of a single property* or *Encryption of multiple properties* section, you need to provide your secret key to EFM at startup. There are two ways to provide the secret key; directly through a property or through a file:

- You can set the secret key directly through a property in the following ways:
 - As a command line argument: `./bin/efm.sh --efm.property.encryptor.password=secretKey`
 - As a Java system property: `-Defm.property.encryptor.password=secretKey`
 - As an Operating System environment variable: `export EFM_PROPERTY_ENCRYPTOR_PASSWORD=secretKey`
 - As a key value pair in the `efm.properties` file: `efm.property.encryptor.password=secretKey`
- You can set the secret key through a file in the following way:

Provide the path to the secret key file in the `efm.property.encryptor.passwordFile` property:

```
efm.property.encryptor.passwordFile=<path-to-file>
```



Note: Please ensure that correct permissions are set on the file so EFM can access it at startup. Also please note that whitespaces are trimmed from the file content.