Anypoint Property Management



Contents

[Introduction 3](#_Toc527066858)

[Properties Syntax 3](#_Toc527066859)

[Mule 3 3](#_Toc527066860)

[Mule 4 4](#_Toc527066861)

[Overview 4](#_Toc527066862)

[Encrypt/Decrypt Properties 4](#_Toc527066863)

[On-Premise (Hybrid) Property Management 5](#_Toc527066864)

[Configuring to Run Locally and on Runtime 6](#_Toc527066865)

[CloudHub Property Management 7](#_Toc527066866)

[Configuring to Run Locally and on Runtime 7](#_Toc527066867)

[Hidden (Masked) Properties 8](#_Toc527066868)

[Make a Hidden Property in Mule 3 8](#_Toc527066869)

[Make a Hidden Property in Mule 4 8](#_Toc527066870)

[General Best Practices 9](#_Toc527066871)

# Introduction

Anypoint applications use configuration properties to handle environment-specific information, such as end-system configurations. Properties are loaded by an app at load time; they are not refreshed while the app remains running. This provides a strategy for managing these properties for Anypoint applications in Cloudhub and on-premise runtimes.

# Properties Syntax

The syntax for secure and non-secure configuration properties changed between Mule 3 and Mule 4. The links and examples below are to the documentation describing the details for each version.

Best Practice: Use yaml files for properties in Mule 4. Mule 3 only uses properties files.

Properties files throughout this document will refer to "yaml" files unless it is a Mule 3 only example.

## Mule 3

* The [ignore-unresolvable="true" attribute](https://support.mulesoft.com/s/article/Secure-property-placeholder-not-working-when-used-together-with-a-regular-property-placeholder) is required for multiple property placeholders in a single app.
* URLs can be used for properties file path in location attribute.
* Use file syntax (file://) when accessing files outside of the classpath.
* Load the common properties file in the domain so it is loaded for all apps a single time. If no domain project used (e.g., Cloudhub), it can be loaded in each app, along with the app's specific properties.

[Non-Secure Properties](https://docs.mulesoft.com/mule-runtime/3.9/properties-configuration-reference)

<context:property-placeholder

location="properties/my-app.properties" ignore-unresolvable="true"/>

[Secure Properties](https://docs.mulesoft.com/mule-runtime/3.9/mule-credentials-vault)

<secure-property-placeholder:config

name="Secure\_Property\_Placeholder"

doc:name="Secure Property Placeholder"

encryptionAlgorithm="AES"

encryptionMode="CBC"

key="${mule.secure.key}"

location="properties/my-app-secure.properties" ignore-unresolvable="true"/>

Secure Properties - Domain

<spring:bean class="org.mule.modules.security.placeholder.SecurePropertyPlaceholderModule">

<spring:property name="encryptionMode" value="CBC" />

<spring:property name="encryptionAlgorithm" value="AES" />

<spring:property name="location" value="${ops.properties.location}/common-secure.properties" />

<spring:property name="ignoreUnresolvablePlaceholders" value="true" />

<spring:property name="ignoreResourceNotFound" value="true" />

<spring:property name="key" value="${mule.secure.key}" />

</spring:bean>

## Mule 4

* Each properties config can load one file. Use multiple configs to load more than one file.
* The file syntax is not needed like it was in Mule 3 when accessing files outside of the classpath.
* Load the common properties file in the domain so it is loaded for all apps a single time. If no domain project used (e.g., Cloudhub), it can be loaded in each app, along with the app's specific properties.

[Non-Secure Properties](https://docs.mulesoft.com/mule4-user-guide/v/4.1/configuring-properties#properties-files)

<configuration-properties

doc:name="Configuration properties"

file="${ops.properties.location}/common-secure.${mule.env}.yaml" />

[Secure Properties](https://docs.mulesoft.com/mule4-user-guide/v/4.1/secure-configuration-properties)

<secure-properties:config

name="Secure\_Properties\_Config"

doc:name="Secure Properties Config"

file="properties/my-app.${mule.env}-secure.yaml"

key="${mule.vault.key}" />

# Overview

Anypoint applications use configuration properties to handle environment-specific information, such as end-system configurations. Properties are loaded by an app at load time; they are not refreshed while the app remains running. This scenario is usually much easier for ops team to manage in prod and pre-prod environments. It also allows for configuration management tools to directly manage the properties in Cloudhub or on-premise.

Secure and regular properties are kept in separate properties files and are imported using the different modules. The regular configuration properties module is used to import the files with all non-secure (plain text) properties. The secure configuration properties module is used to import the file with secure (encrypted) properties.

For encryption, the default and recommended encryption algorithm and mode is AES and CBC respectively though others are available.  Blowfish is often used since it doesn’t restrict the number of characters in the security key, though this is only recommended for training environments. In the lower environments (Development and Test), the developers can encrypt the values on their own in their Mule application projects. However, in the higher environments (Stage and Production), the sensitive credentials will not be known to developers.

# Encrypt/Decrypt Properties

There are several ways to encrypt and decrypt properties externally from a Mule application. This is necessary to create encrypt properties to add them to secure properties files and to decrypt properties from secure properties files to view contents for troubleshooting. Some options can be scripted or automated as necessary for Dev/Ops or CI/CD.

These options are listed below.

* Command-line utility: [Secure Properties tool](https://docs.mulesoft.com/mule-runtime/4.1/secure-configuration-properties#secure-properties-tool) provided by MuleSoft.
* User must know vault key. Requires vault key as input.
* Encrypt & decrypt.
* Single property or complete file per operation.
* Mule application: create a custom API that uses the [Mule Message Encryption processor](https://docs.mulesoft.com/mule-runtime/3.9/mule-message-encryption-processor#jce-encrypter) (Mule 3) or [Cryptography processor](https://docs.mulesoft.com/mule-runtime/4.1/cryptography-jce) (Mule 4) to encrypt/decrypt.
* User does not know vault key. Keeps key in secure keystore per environment.
* Encrypt & decrypt.
* Multiple properties per operation.
* Anypoint Studio: per property in each application and only available in Studio 6 (Mule 3).
* User must know vault key. Requires vault key as input.
* Encrypt & decrypt.
* Single property per operation.

A sample properties file with encrypted values is shown below.

|  |
| --- |
| user: "![DoW5oa21B8GbKDtk+i7yhg==]"  password: "![PMIXPa2XNUbkBfHycwpmgQ==]" |

Best Practice

Create an Encryption API for developer and operator self-service so they can create properties files for specific environments without knowing the vault keys. This API must only be encryption and not decryption. The CI/CD system can also use this instead of the command-line utility if the system does not have access to the vault key.

Create a Decryption API for production operators to decrypt properties for troubleshooting or verifying production properties. This API must be secured and granted access separately from the Encryption API.

The APIs can be in the same Mule app but using different API Kits and RAML so they are different APIs in API Manager.

The APIs can be directly accessed via Postman, a custom web page, or script in the case of the CI/CD system.

# On-Premise (Hybrid) Property Management

The recommended practice for managing properties for on-premise Anypoint runtimes, is to externalize the files from the applications and manage them directly on the runtime’s file system. This allows configuration management systems to manage the properties outside of an app's deployment package. There are two main options for externalizing properties in an on-premise runtime:

* Add and maintain properties files in %MULE\_HOME%/conf folder.
* Add and maintain properties files at a specified location reachable from the Mule server, which is defined by an environment variable.

The recommended option is to set the properties files in their own folder in the conf folder: %MULE\_HOME%/conf/properties. This location is then defined by an environment variable set in wrapper.conf, such as ops.properties.location. The location is a subfolder in the classpath, so this environment variable is technically not needed, but it will make accessing the properties file much easier when testing locally as will be shown later.

Developers need to specify all file locations in the secure and non-secure properties configurations in each of their Mule applications.  This is best done in a global config file. There are two types of properties files put in this location:

* common.yaml: a single file per runtime that contains all the properties that are common across the applications on the runtime. This can include reused configurations for external systems, polling and retrying configurations, and hostname/port for accessing other Anypoint APIs.
* common-secure.yaml: a single file per runtime that contains all the secure properties that are common across the applications on the runtime. This is usually the credentials for the shared systems in the non-secure file.
* <application name>.yaml: a single file per application that contains properties specific to that application. Domain (shared resource) projects also fit this pattern.
* <application name>-secure.yaml: a single file per application that contains secure properties specific to that application. Domain (shared resource) projects also fit this pattern.

Mule 3 Note: Domain projects may share properties loaded by any of their applications to all the applications, with the last loaded property taking precedence. This means that application-specific property names in their own file MUST still be UNIQUE across ALL properties files or the properties will be overwritten by the last loaded application. Recommended practice in this scenario is to add the app name as a prefix to the property name. This only applies to applications that use a shared domain project in Mule 3.

## Configuring to Run Locally and on Runtime

The properties files locations running in Anypoint Studio and on a runtime should work with a single configuration and set with the environment variables. The paths for each file type is detailed below.

**common.properties**

* Config properties path: ${ops.properties.location}/common.properties
* Local project location: external from application project; specific to each system. Can be retrieved from Git or a configuration management system.
* Runtime location: %MULE\_HOME%/conf/properties/common.properties

**<app>.properties**

* Config properties path: properties/my-app.properties
* Local project location: ./src/main/resources/properties/my-app.properties
* Runtime location: %MULE\_HOME%/conf/properties/my-app.properties

# CloudHub Property Management

The recommended practice for managing properties for Anypoint apps deployed to Cloudhub, is to internalize the files into each the application. This is necessary since each app runs in its own VM and don't have access to a common file system.

Mule 4 supports both .yaml and .properties files for properties. Yaml files are preferred over properties files. See the [documentation](https://docs.mulesoft.com/mule4-user-guide/v/4.1/mule-app-properties-to-configure).

Properties files should be put in a folder named properties under the resources folder: ***./src/main/resources/properties***.

## Configuring to Run Locally and on Runtime

The properties files locations running in Anypoint Studio and on a runtime should work with a single configuration and set with the environment variables. Since Cloudhub apps must have properties files embedded, these files need to have the environment in their names so the proper file is loaded per environment.

There should be at least two property files per environment:

* regular properties: <environment>.yml
* secure properties: <environment>-secure.yml

If you need to break properties up in multiple files per environment, use the pattern *<name>-<environment>.yml*. At the minimum, a project should have the files below under ./src/main/resources/properties:

* local.yml
* local-secure.yml
* dev.ym
* dev-secure.ymll
* test.yml
* test-secure.yml
* stage.yml
* stage-secure.yml
* prod.yml
* prod-secure.yml

## Hidden (Masked) Properties

Hidden properties are non-readable properties (masked) for an app, which are described in [Safely Hiding Application Properties](https://docs.mulesoft.com/runtime-manager/secure-application-properties). This feature is only available to apps deployed in Cloudhub. A quick summary of hidden property features is listed below.

* The property is visible in the app's Properties tab.
* The property's value is masked in the app's Properties tab and can never be read.
* The property's value can be changed in the app's Properties tab by an operator.
* The property is stored as an encrypted value in a CloudHub database (encrypted per user org).
* The property is plain text in properties or YAML files if provided in those files. [[1]](#footnote-1)
* Can be provided to the app via:
* Operator inputting in app's Properties tab in Runtime Manager.
* CI/CD specifying properties when deploying app to CloudHub.
* Properties or YAML filesin app package at time of deployment.

The main difference between hidden properties and encrypted properties on CloudHub is that the properties encryption key is a hidden property in an app, while all the passwords and such are all encrypted properties that do not show up in the properties tab in CloudHub.

Best Practice: Always make sure the Mule Vault Key, Anypoint Platform Client ID and Client Secret for the API are hidden properties when deploying to Cloudhub.

### Make a Hidden Property in Mule 3

Follow the steps below to make a property hidden in a project.

1. Open “mule-app.properties” in the Mule project.
2. Add the names of all the hidden properties to the "secure.properties" property.

* Names are separated by commas.
* Example:

secure.properties=anypoint.platform.client\_id,anypoint.platform.client\_secret,mule.vault.key

1. Save the file.

### Make a Hidden Property in Mule 4

Follow the steps below to make a property hidden in a project.

1. Open “mule-artifact.json” in the Mule project.
2. Add the names of all the hidden properties to the secureProperties array.

* Each name must be enclosed in quotes.
* Names are separated by commas.
* Example:

"secureProperties": ["anypoint.platform.client\_id","anypoint.platform.client\_secret", "mule.vault.key"],

1. Save the file.

# General Best Practices

The best practices regarding properties are listed below.

* Keep all sensitive information in encrypted properties and non-sensitive information as unencrypted properties. Keep the different types stored in separate files. This makes automatic generation using the different utilities easier.
* The Mule Credentials Vault encryption key, mule.vault.key, is the encryption key for the environment to decrypt the secure properties:
* Cannot be an encrypted property as there would be no way to decrypt properties.
* Should be a hidden property when in Cloudhub.
* Should be the only non-encrypted hidden property. It will be stored as encrypted in the CloudHub property database.
* Should not be in a properties file in an app.
* Should not be kept in source control.
* Should be provided at deployment time directly by CI/CD system to ARM when deploying an app.
* There is one encryption key per environment.
* Use comments in properties files to describe groupings of properties.
* Use yaml files instead of properties files for properties in Mule 4.
* Cloudhub: Ensure that API and mule.vault.key are always set as hidden properties in mule-artifact.json.  
  "secureProperties": ["anypoint.platform.client\_id","anypoint.platform.client\_secret", "mule.vault.key"]
* Cloudhub: To update encrypted properties without an app redeploy, make the encrypted property a hidden property also.

1. *A property can be a combined Hidden and Encrypted property. In that case, it would take on both feature-sets making the value encrypted and hidden and stored in the encrypted CloudHub database.* [↑](#footnote-ref-1)