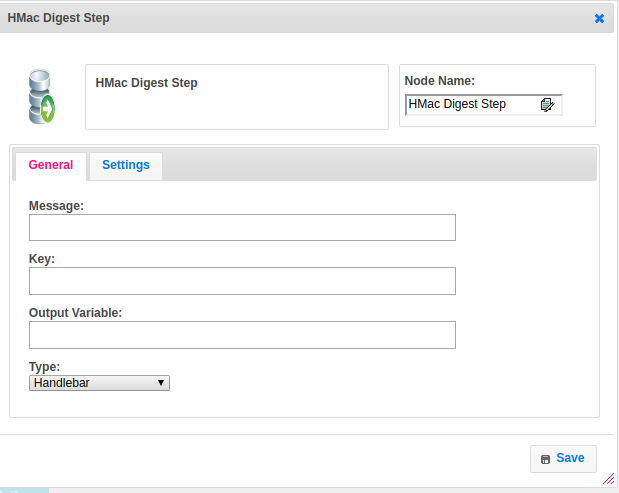
HMAC Message Authentication Code Step

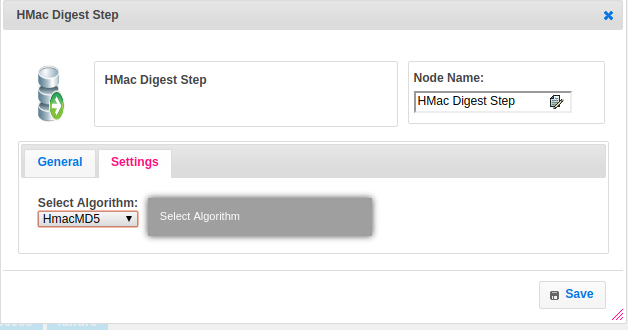
Objective:

A [Message Authentication Code](https://en.wikipedia.org/wiki/Message_authentication_code) or a MAC provides a way to guarantee that a message (a byte array) has not been modified in transit. It is similar to a [message digest](https://en.wikipedia.org/wiki/Cryptographic_hash_function) to calculate a hash, but uses a secret key so that only a person with the secret key can verify the authenticity of the message.

Using a MAC to ensure safe transmission of messages requires that the two parties share a secret key to be able to generate and verify the MAC. There are two approaches available here – the two parties can share a secret key directly. Or the secret key can be generated using a password. We investigate both approaches below.

UI

****

****

Attributes:

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| message | Message for which will send to destination |
| key | Which you can decode and encode the message based on algorithm and it will be sent to receiver along with message |
| Algorithm | Algorithm property is used to converting message by using different types of algorithm **Ex.HMacMD5, HMacSHA1** |
| Value\_type | To select type as free marker template or handlebar |

|  |  |
| --- | --- |
| output\_variable | To store the response of the request for reference |

## Executor Description:

1. **It will read the algorithm from the user for MAC**
2. **SecretKeySpec will acquire key and algorithm and generate the hash**
3. **Then init the mac object by passing keySpec object**
4. **mac.doFinal() is used to convert into mac hashing**

## Dependent Plugins

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Key | Key is nothing but **Client ID**.  The client id is a public identifier for apps. Even though it’s public, it’s best that it isn’t guessable by third parties, so many implementations use something like a 32-character hex string. It must also be unique across all clients that the authorization server handles. If the client ID is guessable, it makes it slightly easier to craft phishing attacks against arbitrary applications. |
| Secret | A secret is nothing but the **Client Secret.**  The client secret is a secret known only to the application and the authorization server. It must be sufficiently random to not be guessable, which means you should avoid using common UUID libraries which often take into account the timestamp or MAC address of the server generating it. A great way to generate a secure secret is to use a cryptographically-secure library to generate a 256-bit value and converting it to hexadecimal representation. |

**SecretKeySpec**. This class specifies a secret key in a provider-independent fashion. It can be used to construct a **SecretKey** from a **byte array**, without having to go through a (provider-based) SecretKeyFactory .

## Resource

More information about HMAC message digest please refer the following link.

https://www.javacodegeeks.com/2012/10/what-is-hmac-authentication-and-why-is.html