# **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.

**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 .

**POJO 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**

Json

{

"workflows": [

{

"name": "hash-func",

"id": 200,

"trigger": "rest",

"expression": "hash/func",

"method": "GET",

"steps": [

{

"id": "1",

"type": "hash-func-digest",

"file\_data": "narahari",

"algorithm": "MD5",

"output\_variable": "digested",

"next": {

"success": "s",

"failure": ""

},

"value\_type": "hb"

},

{

"id": "s",

"type": "send",

"variable\_name": "digested"

}

]

}

],

"filters": [],

"server": {},

"plugins": []

}

// The Java Mac (javax.crypto.Mac class can create a Message Authentication Code (MAC) from binary data.

Mac mac = Mac.*getInstance*(step.getAlgorithm());

// Constructs a secret key from the given byte array, using the first len bytes of key, starting at offset inclusive.

SecretKeySpec keySpec = new SecretKeySpec(step.getKey().getBytes(), step.getAlgorithm());

mac.init(keySpec);

byte[] mac\_data = mac.doFinal(step.getMessage().getBytes(StandardCharsets.*UTF\_8*));

map.put(step.getOutput\_variable(), StringUtil.*convertToHex*(mac\_data));