



Dealing with Secured Target Servers

Building Target Authentication

1. Hard Code

Building Target Authentication

1. Hard Code
2. Replace at Build Time

Building Target Authentication

1. Hard Code
2. Replace at Build Time
3. Node.js Vault

Building Target Authentication

1. Hard Code
2. Replace at Build Time
3. Node.js Vault
4. Encrypted Key Value Map

Hard Code

```
<AssignMessage name="Set-Credentials">
  <DisplayName>Set-Credentials</DisplayName>
  <Properties/>
  <Set>
    <Headers>
      <Header name="Authorization">Basic Secret123==</Header>
    </Headers>
  </Set>
  <AssignVariable>
    <Name>username</Name>
    <Value>secretUser</Value>
  </AssignVariable>
  <AssignVariable>
    <Name>password</Name>
    <Value>secretPassword</Value>
  </AssignVariable>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

Build Time Replacement

```
<AssignMessage name="Set-Credentials">
  <DisplayName>Set-Credentials</DisplayName>
  <Properties/>
  <Set>
    <Headers>
      <Header name="Authorization">Replace Me</Header>
    </Headers>
  </Set>
  <AssignVariable>
    <Name>username</Name>
    <Value>replace_me</Value>
  </AssignVariable>
  <AssignVariable>
    <Name>password</Name>
    <Value>replace_me</Value>
  </AssignVariable>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

Build Time Replacement

```
{
  "configurations":[
    {
      "name":"test",
      "policies":[
        {
          "name":"Set-Credentials.xml",
          "tokens":[
            {
              "xpath":"/AssignMessage/Set/Headers/Header[@name='Authorization']",
              "value":"Basic Secret123=="
            },
            {
              "xpath":"/AssignMessage/AssignVariable[Name='username']/Value",
              "value":"secretUser"
            },
            {
              "xpath":"/AssignMessage/AssignVariable[Name='password']/Value",
              "value":"secretPassword"
            }
          ]
        }
      ]
    }
  ]
}
```


Build Time Replacement

```
<AssignMessage name="Set-Credentials">
  <DisplayName>Set-Credentials</DisplayName>
  <Properties/>
  <Set>
    <Headers>
      <Header name="Authorization">Basic Secret123==</Header>
    </Headers>
  </Set>
  <AssignVariable>
    <Name>username</Name>
    <Value>secretUser</Value>
  </AssignVariable>
  <AssignVariable>
    <Name>password</Name>
    <Value>secretPassword</Value>
  </AssignVariable>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

Using Node.js Access Vault

Step 1: Build Vault Via API

```
curl https://api.enterprise.apigee.com/v1/o/{org}/vaults  
-H "Content-Type: application/json"  
-d '{"name": "vault1" }' -X POST
```

Using Node.js Access Vault

Step 1: Build Vault Via API

```
curl https://api.enterprise.apigee.com/v1/o/{org}/vaults  
-H "Content-Type: application/json"  
-d '{"name": "vault1" }' -X POST
```

Step 2: Create the entries within the Vault Via API

```
curl  
https://api.enterprise.apigee.com/v1/o/{org}/vaults/{vault}/ent  
ries  
-H "Content-Type: application/json"  
-d '{"name": "username", "value": "secretUser" }' -X POST
```

Using Node.js Access Vault

Step 1: Build Vault Via API

```
curl https://api.enterprise.apigee.com/v1/o/{org}/vaults
-H "Content-Type: application/json"
-d '{"name": "vault1" }' -X POST
```

Step 2: Create the entries within the Vault Via API

```
curl
https://api.enterprise.apigee.com/v1/o/{org}/vaults/{vault}/ent
ries
-H "Content-Type: application/json"
-d '{"name": "username", "value": "secretUser" }' -X POST
```

Step 3: Retrieve Data in Node.js App

```
var apigee = require('apigee-access');
var orgVault = apigee.getVault(vault1, 'organization');
orgVault.get('username', function(err, secretValue) {
  // use the secret value here
});
```

Using Key Value Map

New Key Value Map

Name

credsMap

Encrypted

☒

Cancel

Add

▼ credsMap

KEY	VALUE
username	*****
password	*****

Using Key Value Map

Step 2: Retrieve Data via KVM Policy

```
<KeyValueMapOperations name="getEncrypted"
mapIdentifier="credsMap">
  <Scope>environment</Scope>
  <Get assignTo="private.encryptedUsername"
index="1">
    <Key>
      <Parameter>username</Parameter>
    </Key>
  </Get>
  <Get assignTo="private.encryptedPassword"
index="1">
    <Key>
      <Parameter>password</Parameter>
    </Key>
  </Get>
</KeyValueMapOperations>
```



Thank You