Gateway Payload Digital Signature

The current payload has the following structure

```
{
"hid"          : "string",
"name"         : "string",
"encrypted"    : "boolean",
"parameters"   :
{
"Key1"     : "string",
"Key2"     : "string",
}
}
```

Two new parameters "signature" and "signatureVersion" are added to this payload. Use the following procedure to derive the signature value

For example in this document, we use the following keys and payload

- apiKey: 5501f50fdc62aee5d04dbd6a58b68b781ee2aaade8ad1eb24b1e4e77cb282ae2
- secretKey: ARAzUzRzekFwRTNACBQYUx89LlZyImhKFVloHUVMDw8EGRxxSCckFgdFPysAAWJCLDgMdkstZzw3GGVqNHxXcno5Iz54LRBSKy0TaCBwNndkfQNdD38KAA==

```
{
"hid"          : "05c2d78dee6798025e6e3f83f79256914b7c3664",
"name"         : "update-configuration",
"encrypted"    : "false",
"parameters"   :
{
"Key1"     : "Value 1",
"Key2"     : "Value 2",
}
}
```

Step 1: Create a Canonical Request for signing

The first step is to build a string representation of your request in a pre-defined format.

```
canonicalRequest =
hid + '\n' +
name + '\n' +
encrypted + '\n' +
canonicalParameterString
```

hid

hid value from payload

name

name value from payload

encrypted

encrypted value from payload

canonicalParameterString

- Create each line containing a name/value pair in the format name=value. Name field is converted to lowercase
- Sort all the lines in ascending order

Example

```
key1=Value 1
key2=Value 2
```

The canonical request will be as follow

```
05c2d78dee6798025e6e3f83f79256914b7c3664
update-configuration
false
key1=Value 1
key2=Value 2
```

Finally, hex-encode the computed canonical request above and lowercase it. This is the result of step 1.

```
fd5a714bd34324574d81df94d7021c12da0a157e3b99a33938140c6a10936e6d
```

Step 2: Create the string to sign

The structure of the string is as follow

```
stringtoSign =
hashedCanonicalRequest + '\n' +
apiKey + '\n' +
signatureVersion
```

hashedCanonicalRequest

This is the result of step 1

```
fd5a714bd34324574d81df94d7021c12da0a157e3b99a33938140c6a10936e6d
```

apiKey

Provided apiKey from Kronos portal

```
5501f50fdc62aee5d04dbd6a58b68b781ee2aaade8ad1eb24b1e4e77cb282ae2
```

signatureVersion

Current version is 1

Combining all of the results above produces the following string. This is the result of step 2.

```
fd5a714bd34324574d81df94d7021c12da0a157e3b99a33938140c6a10936e6d
5501f50fdc62aee5d04dbd6a58b68b781ee2aaade8ad1eb24b1e4e77cb282ae2
1
```

Step 3: Create the signing key

In this step we're going to use the HMAC-SHA256 algorithm a few times to generate the signing key. In the pseudocode, it'll be shown as hmacSha256(key, data). Note the order of the input parameters because if you're using some third-party library for this function, the order of the input parameters might be reversed. The output of this function is in binary, hence note the hex-encode function being used below

```
signingKey = secretKey
signingKey = hexEncode(hmacSha256(apiKey, signingKey))
signingKey = hexEncode(hmacSha256(signatureVersion,
signingKey))
```

secretKey

Provided secretKey from Kronos portal

apiKey

Provided apiKey from Kronos portal

signatureVersion

Must be the same value as in step 2

For the example in this guide, here's the output of this step. This is the result of step 3

```
signingKey =
ARAzUzRzekFwRTNACBQYUx89LlZyImhKFVloHUVMDw8EGRxxSCckFgdFPysAAWJCLDgMdkstZzw3GGVqNHxXcno5Iz54LRBSKy0TaCBwNn
signingKey = 3c6e85f6a719e5b8bd77fde0cbdbe19d947f38451afbc8ef6e49a083d86a9c54
signingKey = 2c25562ec92ac4e6f52449c3c34ce8d860578372af1b958656790a47d4b76093
```

Step 4: Create the final signature

The signature is computed by signing the result from step 2 and step 3

```
signature = hexEncode(hmacSha256(signingKey,
stringToSign)
```

signingKey

Result of step 3

stringToSign

Result of step 2

Result of the current example is

```
2bcc72adcef72780dfd436d4de46054a49f6bcb832dc2bd3ec05a54da275b8b5
```

The final payload would be

```
{
"hid"         : "05c2d78dee6798025e6e3f83f79256914b7c3664",
"name"        : "update-configuration",
"encrypted"   : "false",
"parameters"  :
{
"Key1"    : "Value 1",
"Key2"    : "Value 2",
}
"signature"   : "2bcc72adcef72780dfd436d4de46054a49f6bcb832dc2bd3ec05a54da275b8b5",
"signatureVersion" : "1"
}
```