COVE API Authentication User Guide

This document describes how to generate the correct signature hash when making COVE API calls

Prerequisites

Any COVE API destination will need some information set up in the COVE API admin tool. The software must be identified and a 'destination' should be created.

With the destination created, there are two key pieces of information that will be created:

1. API ID
2. API Secret

The API ID will be the unique identifier for the destination. Any software that wishes to use the COVE API should be uniquely identified by this value.

The API Secret is a value that will be used in the signing process. It should not be displayed or made public.

The client must also have the ability to create a HMAC using the SHA-1 algorithm.

Establishing the Request

The appropriate queries on the API should be developed using the API User Guide. The client should construct any queries to meet their needs.

Creating the signature

In order to generate the signature the client will need the following pieces of information:

- **timestamp** - The current timestamp in the form of number of sections since Jan 1st, 1970 00:00:00 UTC
- **API ID** - As defined above
- **API Secret** - As defined above
- **nonce** - A randomly generated string. Chosen by the client, this value prevents 3rd parties from sniffing the API queries on the network and using that information to impersonate or fake additional queries. The values allowed to be chosen include all lower and uppercase English characters (a-z and A-Z) and the special character dash ('-').
- **verb** - The HTTP verb intended to be used by the client. Generally either 'GET' or 'POST'
- **uri** - The request created in the previous step.

All these values will be processed to create a new value: **signature**. All these values will be combined to create a signed COVE API request.

Generate a canonical URI

There are different interpretations of 'canonical,' but in this case this means:
`<protocol>://<hostname>/<query string>?<param1>=<key1>[&<param2>=<key2>...]`
Additionally, the parameters should be listed in alphabetical order.

So take the **uri** query in full as generated above and add the following parameters:

```
&consumer_key=<API ID>
&nonce=<nonce>
&timestamp=<timestamp>
```

And then sort all the parameters alphabetical. Ensure this is in UTF-8 format (which generally means no conversion for common English characters). Also ensure this is url decoded (e.g. Spaces are not represented as %20 but as single ' ' characters).

Example
Given the request query:

http://api.pbs.org/cove/v1/videos?format=json&filter_nola_root=NOVA&filter_t

And a timestamp of 12345, a nonce of 'abcdef-tuv-wxyz', and a consumer_key of 'test-abc-123'.
The canonical URI is:

http://api.pbs.org/cove/v1/videos?consumer_key=test-abc-123&filter_nola_root

or to allow for word-wrap in this document:

```
http://api.pbs.org/cove/v1/videos?
   consumer_key=test-abc-123
   &filter_nola_root=NOVA
   &filter_type=Episode
   &format=json
   &nonce=abcdef-tuv-wxyz
   &timestamp=12345
```

Note the parameters are sorted and there are no url encodings

Generate a string to be signed

The string that will be signed comprises the following items with no spaces between:

HTTP verb + Canonical URI + Request Body + timestamp + consumer_key + nonce

Example
To continue our example from above the string would be

```
GEThttp://api.pbs.org/cove/v1/videos?consumer_key=test-abc-123&filter_nc
```

or to allow for word-wrap at 40 characters:

```
GEThttp://api.pbs.org/cove/v1/videos?con
sumer_key=test-abc-123&filter_nola_root=
NOVA&filter_type=Episode&format=json&non
ce=abcdef-tuv-wxyz&timestamp=1234512345t
est-abc-123abcdef-tuv-wxyz
```

Calculate HMAC using SHA-1

We'll use SHA-1 as the hash algorithm. The signature should be output in hex mode.
Most implementations of HMAC will take a key and a string. The key used should be the API Secret in te

Assuming the API Secret is '843e62bafd4573263e439a2463b4fe78b9a0b14c' the signature for the string a
`3231b9c2b2f247d31aa8bc6495615e0ad8f8b665`

Full Example

Assume a destination has the following settings:

```
API ID: JOSHTEST-171ca88c-e987-4a06-a3f7-d46f2d6fa272
API Secret: 1e7321e3-30f5-4df1-82a0-c4dd283b5f65
```

And the values:

```
timestamp=1288144873
nonce=c21d32917b0e71febd9
```

Query1

```
Request: http://api.pbs.org/cove/v1/videos/?filter_nola_root=SOTM&filter_mediafi
Canonical URI: http://api.pbs.org/cove/v1/videos/?consumer_key=JOSHTEST-171ca88c
Signed String:
GEThttp://api.pbs.org/cove/v1/videos/?consumer_key=JOSHTEST-171ca88c-e987-4a06-a
Signature:
e3004de2e2dd45604136262fa31a06217f72e87b
```

Useful tools

There is an online HMAC tool that leverages SHA-1 at: http://hash.online-convert.com/sha1-generator
This can be used to test implementation of this algorithm.

Flash Libraries

To implement in Action Script, the following library can be used:

http://code.google.com/p/as3crypto/

PHP Libraries

- ⚠ TBD

Python Libraries

- ⚠ TBD

Objective-C Libraries

- ⚠ TBD