WORLD
SYNC

1WorldSync Content1 Web Services

API HMAC Guide

Version 1.1

REVISION HISTORY

Date	Ver #	Description of Change	Author
October 14, 2015	1.0	Initial Version	1WorldSync
October 26, 2016	1.1	Updated links	1WorldSync

TABLE OF CONTENTS

1 Overview

1WorldSync's Content1 ContentNOW API allows you to access 1WorldSync's rich and trusted repository of Brand Owner provided product information.
This document describes how to use the APIs to access product data from 1WorldSync Content1. It details the various parameters required to access and use the service, such as web service URLs, and request and response formats.

The intended audience of this document is anyone who will use Web services to access ContentNOW product data from Content1. (Note that a separate API document exists for those subscribers that will provide digital asset content.)

The current application programming interface (API) is a version 1 (/V2/) release.

These Web services will help you to:

- Access trusted product data sourced directly from Brand Owners and Manufacturers to power your applications.
- Ensure that you do not miss out on exciting information about new products from leading brands.
- Access latest updates on products, real-time.
- Perform efficient and extremely fast search operations on products.

2 If you are new to 1WorldSync's Content1 API

Before using 1WorldSync's web services, you must first register at https://developer.1worldsync.com to receive the credentials required for access.

Upon registration, you'll receive the following credentials:

- Your app_id: a 8-character, alphanumeric identifier
- Your X-3SCALE-AUTH-SECRET: a 32-character identifier

The app_id must be included in all 1WorldSync Content1 web service requests to identify the sender of the request. The X-3SCALE-AUTH-SECRET code allows you to create the digital signature that provides proof that you truly are the sender of the request. For all requests you must calculate this signature (hash_code) using your X-3SCALE-AUTH-SECRET code to authenticate your signature. You may refer to Appendix B: Authorization Mechanism to Access 1WorldSync API for more details on the information you must supply for the authentication process.

If you would like to test our API's in a non-production environment, you can use our Pre-Production environment: https://marketplace.preprod.api.1worldsync.com/V2/products. You will need a separate client id and secret key from your production account; contact us at customersupport@1worldsync.com if you need this additional access established.

Otherwise access to the production web services will be through:
https://marketplace.api.1worldsync.com/V2/products

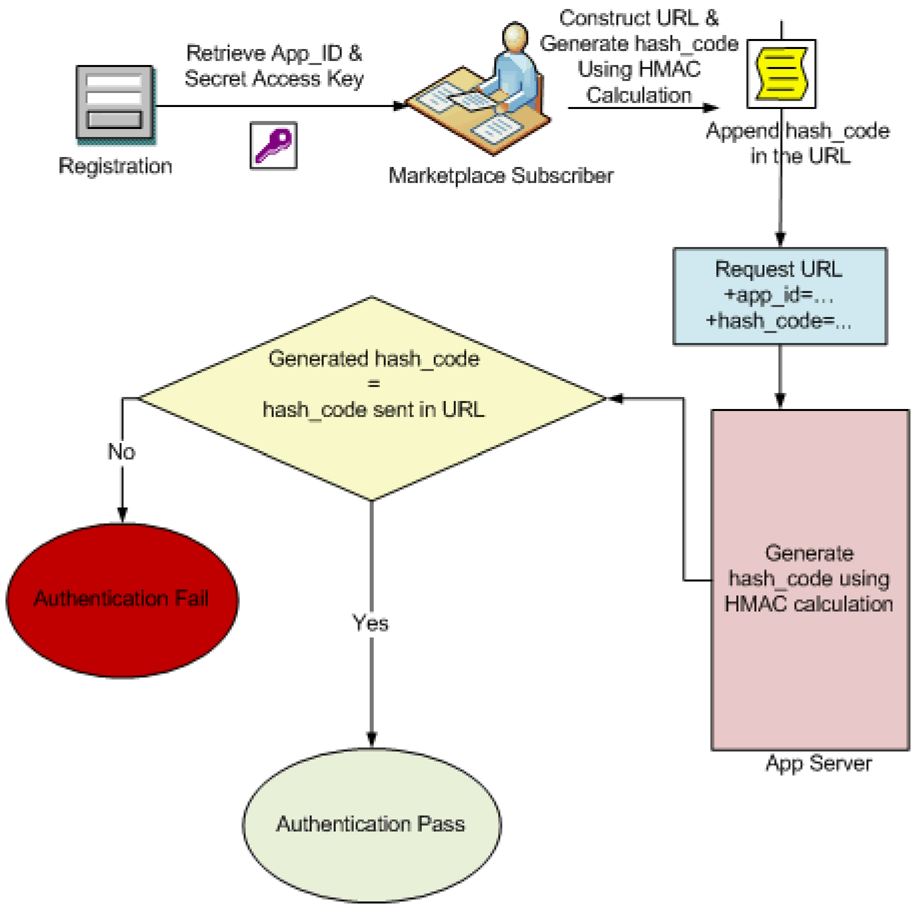
There is also an online toolkit and sdk available to help you begin programming against the Content1 API while allowing you to make direct API calls using your app_id and secret key. Visit https://marketplace.api.1worldsync.com/api/ for more information.

3 Authorization Mechanism to Access Web Services

When accessing 1WorldSync web services using REST API, a hash code must be supplied in order for 1WorldSync to authenticate your request. Following are the steps for incorporating a hash code in your web services request:

1. A new Marketplace user will register at https://developer.1worldsync.com portal. Post-registration, the user is provided with two keys.
   - app_id
   - X-3SCALE-AUTH-SECRET

2. Construct an API request with the parameters mentioned in under **Request Parameters** in sections searching for Information and Fetching Product Information.
3. Calculate a keyed-hash message authentication code (HMAC-SHA256) signature (i.e. hash_code) using the X-3SCALE-AUTH-SECRET for this request URL.
4. Include the hash_code and app_id parameter with value into the request and send the request to 1Worldsync webserver.



Please find the code snippet for HMAC calculation and Encoding:

```
public static String calculateRFC2104HMAC(String data, String key)
throws java.security.SignatureException, java.security.NoSuchAlgorithmException,
java.security.InvalidKeyException, UnsupportedEncodingException {
String result;
// get an hmac_sha1 key from the raw key bytes
javax.crypto.spec.SecretKeySpec signingKey = new
javax.crypto.spec.SecretKeySpec(key.getBytes(),
"HmacSHA256");

// get an hmac_sha1 Mac instance and initialize with the signing key
javax.crypto.Mac mac = javax.crypto.Mac.getInstance("HmacSHA256");
mac.init(signingKey);

// compute the hmac on input data bytes
byte[] rawHmac = mac.doFinal(data.getBytes());

// base64-encode the hmac
result = org.apache.commons.codec.binary.Base64.encodeBase64String(rawHmac);
result = java.net.URLEncoder.encode(result.trim(),"UTF-8");
return result;
}
```

Below mentioned code snippet can be used to generate the current time stamp

```
       private String getTimeStamp(String offset) {

          String ISO_FORMAT = "yyyy-MM-dd'T'HH:mm:ss";
          SimpleDateFormat isoFormatter = new SimpleDateFormat(ISO_FORMAT);
          String date = null;
          String timeStamp= null;

          if(offset==null) {

             isoFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));
             date = isoFormatter.format(new Date());
             timeStamp = date + "Z";
          } else {

             try {

                   int offsetProvided=
                         (Integer.parseInt(offset.substring(1,3)))*60
                         +(Integer.parseInt(offset.substring(4,6)));

                   char offsetSign =offset.charAt(0);
                   if(offsetSign=='-'){
                         offsetProvided=-offsetProvided;
                   }

                   int offsetSystem=
                         (Calendar.getInstance().getTimeZone().getRawOffset())/60000;

                   offsetProvided=-offsetProvided+offsetSystem;
                   offsetSystem=offsetProvided/60;
                   offsetProvided=offsetProvided%60;
                   Calendar cal = Calendar.getInstance();
```

```
                cal.add(Calendar.HOUR_OF_DAY, (-offsetSystem));
                cal.add(Calendar.MINUTE, (-offsetProvided));

                timeStamp=isoFormatter.format(cal.getTime())+offset;

        } catch(Exception e) {

                logger.info("Please provide offset in valid format");
                isoFormatter.setTimeZone(TimeZone.getTimeZone("UTC"));
                date = isoFormatter.format(new Date());
                timeStamp = date + "Z";
        }
}
```

How to generate a URL?

An example URL.
scheme://host:port/context/resourcePath?arg1=val1&arg2=val2....&argN=valN

A truth table of which elements should be URL encoded and which elements should be considered into the Hash is presented below.

1. **scheme** => Does not need to be URL encoded. This should not be considered into the Hash-code.

2. **host** => Does not need to be URL encoded. This should not be considered into the Hash-code.

3. **port** => Does not need to be URL encoded. This should not be considered into the Hash-code.

4. **context** => Does not need to be URL encoded since 1WorldSync has taken care not to introduce any reserved characters into the context. However this must be considered into the Hash-code.

5. **resourcePath** => Does not need to be URL encoded since 1WorldSync has taken care not to introduce any reserved characters into the resourcePath. However this must be considered into the Hash-code.

6. **queryParam** (arg1 from above) => Does not need to be URL encoded since 1WorldSync has taken care not to introduce any reserved characters into the context. However this must be considered into the Hash-code.

7. **queryParamValue** (val1 of arg1 from above) => Each param value needs to be individually URL encoded in UTF-8 before appending to the URL. This must also be considered into the Hash-code.

The URL Construction Algorithm

Note:
1. **||** is used to designate the string concatenation operator in this pseudo-code.

2. **utctimestamp** should be time in the UTC form. ex: 2015-10-19T09:58:37Z. See parameters section for a complete description.

3. **urlenc** denotes URL Encoding in UTF-8.

The string to be hashed needs to be prepared as below
HASH = /context/resourcePath || '?' || arg1 || = || val1 || & || arg2 || = || val2 || & || 'app_id' || = || app_id & || 'TIMESTAMP' || = || utctimestamp

HASHCODE = create a hashcode using the HASH string generated above and your secureKey. See the code snippets given in guide on how to create a Hashcode.

The URL to be generated is as below.
URL = scheme://host:port/context/resourcePath
URL = URL || ? || arg1 || = || urlenc(val1) || & || arg2 || = || urlenc(val1) || & || 'app_id' || = || app_id & || 'TIMESTAMP' || = || urlenc(utctimestamp) || & || 'hashcode' || = || urlenc(HASHCODE)

Example URLs;

For an app_id=9af172d4 with a secretKey = XXXXX
an example advanced search request to the URL
https://marketplace.api.1worldsync.com/V2/products
with arguments
access_mdm=COMPUTER
geo_loc_access_long=51.51
geo_loc_access_latd=9.91
query=itemPrimaryId:00007252147019
searchType=advancedSearch
should be handled as below

The string to be hashed for the above request will be as below. As you can see, the string to be hashed is in a un-encoded form and starts from the context /V1.

/V2/products?app_id=9af172d4&searchType=advancedSearch&query=itemPrimaryId:A00007252147019&access_mdm=computer&TIMESTAMP=2015-10-19T09:58:37Z&geo_loc_access_latd=9.91&geo_loc_access_long=51.51

The URL encoded hash-code generated from the above will be =>
g46a7iec6G8lEegcSNYcgiOyFMp0o6YWXyc1sn8YXW0%3D

The final URL generated to make a request will be

https://marketplace.dev.api.1worldsync.com/V2/products?app_id=9af172d4&searchType=advancedSearch&query=itemPrimaryId%3A00007252147019&access_mdm=computer&TIMESTAMP=2015-10-19T09%3A58%3A37Z&geo_loc_access_latd=9.91&geo_loc_access_long=51.51&hash_code=g46a7iec6G8lEegcSNYcgiOyFMp0o6YWXyc1sn8YXW0%3D

4 Contact Us

In case you face any problem, please reach our Customer Support at CustomerSupport@1worldsync.com, or, call any one of the following phone numbers.

Phone	From within USA	From outside USA
Global Customer Support	+1 866.280.4013	+1 312 463 4467