Ad-ID CEA Services API Guide

Updated Oct 26, 2020

Contents

Getting Started

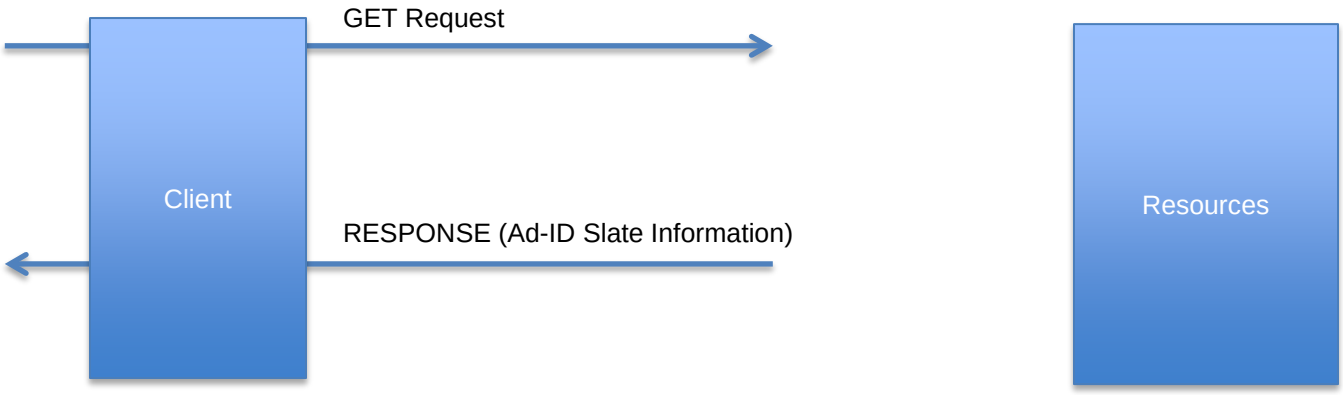
To obtain access to Complete External Access (CEA), please contact Harold Geller (hgeller@ad-id.org) or Raul Rodriguez (rrodriguez@ad-id.org) for a licensing agreement. Once Terms & Conditions are accepted and approved by Ad-ID, you will be provided a user id and authentication key for access to the service.

Overview

The CEA services API supports REST requests for information about Ad-IDs and associated metadata. CEA REST requests are simple HTTP requests, using the GET method with parameters in the URL, and custom HTTP headers. The response may be an XML or JSON document that conforms to a schema, an HTML page, or an HTML snippet (i.e. data in HTML format that is not a complete page).

Using the interface, Ad-ID clients can retrieve Ad-ID Slate information for a given Ad-ID (ADID) or Ad-ID Compact Unique Identifier (CUID).

CEA RESTful Model



CEA Requests

Here, using sample data, is a hypothetical path component of a CEA adid data request, and the three required custom headers:

GET /adid_services/ea_c/adid/ADID0001000

X-Userid: A8U978X0
X-Date: 2015-10-08T10:00:00-04:00
X-Hash: 7cc4d54522a2b45835c14b4fa87a7e7adaaa503452ab38443e3bf55eb0d94a70

URI Format

CEA supports two types of service requests (validation and data), two identifier parameters (Ad-ID or compact identifier), and four response formats (XML, JSON, full HTML page, and HTML snippet). The default format is XML.

The following requests will return XML.

- Validation
  - URI using Ad-ID: /adid_services/ea_v/adid/ADID0001000
  - URI using cuid: /adid_services/ea_v/cuid/abf6cda3
- Data
  - URI using Ad-ID: /adid_services/ea_c/adid/ADID0001000
  - URI using cuid: /adid_services/ea_c/cuid/abf6cda3

A format query parameter and optional type parameter (HTML only) are required to receive a response in HTML or JSON format. The default HTML response type is full; the type parameter may be omitted if HTML format is specified and a full page is desired. HTML responses are not defined for validation requests.

- Data (HTML)
    - URI using Ad-ID: /adid_services/ea_c/adid/ADID0001000?format=html&type=full
    - URI using cuid: /adid_services/ea_c/cuid/abf6cda3?format=html&type=full
    - URI using Ad-ID: /adid_services/ea_c/adid/ADID0001000?format=html&type=snippet
    - URI using cuid: /adid_services/ea_c/cuid/abf6cda3?format=html&type=snippet

- Data (JSON) - Note that "type" is not used for JSON formats.
    - URI using Ad-ID: /adid_services/ea_c/adid/ADID0001000?format=json.
    - URI using cuid: /adid_services/ea_c/cuid/abf6cda3?format=json

CEA Demo Server

Ad-ID provides a server for clients to test their CEA implementations. The URL for this server is https://demo-cea.ad-id.org. Note that this demo server has its own set of Ad-ID codes. Users with accounts on demo.ad-id.org may request any codes they have created on that server. Otherwise for your convenience we provide the following example test codes:

ADID0011000
ADID0012000
ADID0013000
ADID0014000
ADID0015000

Authentication

The CEA service authenticates requests. All HTTP traffic for CEA will take place over SSL. Authentication to the CEA service requires the generation of a HMAC signature using the path component of the URL, combined with a date/time value in ISO 8601 (RFC3339) format.

Overview

1. A user obtains a user ID and API Key, usually upon completing the CEA service agreement. These credentials are normally provided once by Ad-ID to a registered user.
2. Credentials are included with each request. The CEA service uses the credentials to verify that the request is from a valid CEA user.
3. If the credentials are valid, the request is processed, and response information is returned. If the credentials are invalid, the request is rejected, and an error message returned.

Credentials

Each CEA account is provisioned with the following credentials:

- CEA User ID – an 8-character alphanumeric sequence, uppercase letters only.

  Example: A8U978X0

- API Key – a 16-character, alphanumeric sequence, upper and lowercase letters.

  Example: 8E68B85B59bAa36e

Authentication Process Flow

The following describes the steps required to authenticate requests to CEA using an HMAC-SHA256 request signature.

1. You construct a request to CEA.
2. You compute a message authentication code (HMAC-SHA256) using your CEA API Key. For information about HMAC, see RFC2104.
3. You include the following custom HTTP headers in the request, and then send the request to CEA.
4. x-userid (CEA user id provided by Ad-ID)
5. x-date (an ISO 8601 format date/time of the request)
6. x-hash (a hashed value of the path component of the uri and the date/time per Ad-ID specifications).
7. CEA uses your user ID to look up your CEA API key.
8. CEA generates a signature from the request data and CEA key with the same algorithm you used to calculate the signature you sent in the request.
9. If the signature generated by CEA matches the one you sent in the request, the request is considered authentic. If the comparison fails, the request is discarded, and CEA returns an error response.

Creating the HMAC

1. Create the string to sign according to the following pseudo-grammar
   StringToSign = HTTPRequestURI + "+" + ISOFormatTimestamp

   The HTTPRequestURI is the HTTP absolute path component of the URI. The ISOFormatTimestamp is the current timestamp formatted according to the ISO 8601 standard. A plus (+) character is placed between the two components, with no surrounding spaces.

2. Calculate an RFC 2104-compliant HMAC with the string you just created, your CEA API key as the key, and SHA256 as the hash algorithm. For more information, see RFC2104. Please note that the API key should be used as-is, and not altered prior to use.

3. Use the HMAC you computed as the value of the x-hash custom HTTP header.

CEA Responses

There are three possible response codes for any given CEA request. They are:

0 : The Ad-ID/GUID was found and the information is contained in the balance of the response
1 : The Ad-ID/GUID does not exist
2 : The Ad-ID/GUID is valid but the information requested is denied. Note: this response is not returned for validation requests.

The default response will be a well-formed XML document similar to the following:

```
<response>
    <status> [status code from above] </status>
    <count> [the number of codes returned in this query, set to 1 or 0 for V1] </count>
    <status_message>[Status Message] </status_message>
    <codes>
        <code>
            <adid> [the adid code] </adid>
```

```
                    <guid> [the adid compact identifier ] </guid>
                    <field 1> … </field 1>
                    <field n> … </field n>
                </code>
        </codes>
</response>
```

Details concerning responses for successful and failed requests follow.

Data Request Examples (XML)

The following are examples of an XML response to a data request.

Response for status 0

```
<?xml version="1.0" encoding="UTF-8"?>
<adids>
      <status>0</status>
      <count>1</count>
      <adid>
       <adid_fullcode>ZADE0001000H</adid_fullcode>
       <guid>fb1a1dfe</guid>
       <slate>
          <media_type>Video</media_type>
          <video_format_flag>H</video_format_flag>
          <parent id="U10000160">AD EYE DEE CORP</parent>
          <advertiser id="C10000161">AD EYE DEE STORES</advertiser>
          <brand id="B10000162">EYEGLASSES</brand>
          <product id="P10000165">REGULAR VISION</product>
          <ad_title>Seeing is Believing</ad_title>
          <created>2015-09-25</created>
          <copyright>2015 Ad Eye Dee Corp</copyright>
          <version>Free case</version>
          <agency_name>Ad-ID, LLC</agency_name>
          <language>English</language>
          <length>30</length>
          <bleed></bleed>
          <color_type></color_type>
          <expandable></expandable>
        </slate>
       <Brand_and_Product>
          <industry_group id="G700">RETAIL</industry_group>
          <major_category id="G710">RETAIL STORES</major_category>
          <sub_category id="G71E">OPTICAL GOODS AND SERVICES</sub_category>
          <product_category id="G71E">OPTICAL CTR</product_category>
       </Brand_and_Product>
       <commercial_delivery>
          <group>Extreme Reach</group>
        </commercial_delivery>
    </adid>
</adids>
```

Response for status 1

```
<?xml version="1.0" encoding="UTF-8"?>
<adids>
   <status>1</status>
  <status_message> The Ad-ID was not found</status_message>
  <count>0</count>
</adids>
```

Responses for status 2

```
<?xml version="1.0" encoding="UTF-8"?>
<adids>
    <status>2</status>
   <status_message>The Ad-ID is valid but has been excluded.</status_message>
    <count>1</count>
    <adid>
        <adid_fullcode>ZADE0001000H</adid_fullcode>
        <guid>fb1a1dfe</guid>
        <parent>XYZ Corporation</parent>
    </adid>
</adids>
```

Access Denied

```
<?xml version="1.0" encoding="UTF-8"?>
<adids>
    <status>2</status>
   <status_message>The Ad-ID has been voided.</status_message>
    <count>1</count>
    <adid>
        <adid_fullcode>ZADE0001000H</adid_fullcode>
        <guid>fb1a1dfe</guid>
    </adid>
</adids>
```

Voided Code – A user removed this code from the list of active Ad-ID codes.

Data Request Examples (JSON)

The following are examples of a JSON response to a data request.

Response for status 0


```
{
   "status":0,
   "status_message":"The Ad-ID is valid",
   "count":1,
   "adid":"ZADE0001000H",
   "guid":"fb1a1dfe",
   "media_type":"Video",
   "video_format_flag":"H",
   "parent":"AD EYE DEE CORP",
   "parent_id":"U10000160",
   "advertiser":"AD EYE DEE STORES",
   "advertiser_id":"C10000161",
   "brand":"EYEGLASSES",
   "brand_id":"B10000162",
   "product":"REGULAR VISION",
   "product_id":"P10000165",
   "ad_title":"Seeing is Believing",
   "created_date":"2015-09-25",
   "copyright":"2015 Ad Eye Dee Corp",
   "version":"Free case",
   "agency_name":"Ad-ID, LLC",
   "language":"English",
   "length":"30",
   "bleed":null,
   "color_type":null,
   "industry_group":"RETAIL",
   "industry_group_id":"G700",
   "major_category":"RETAIL STORES",
   "major_category_id":"G710",
   "sub_category":"OPTICAL GOODS AND SERVICES",
   "sub_category_id":"G71E",
   "product_category":"OPTICAL CTR",
   "product_category_id":"G71E",
   "external_groups":"Extreme Reach"
}
```


Response for status 1


```
{
  "status":1,
  "message":"The Ad-ID was not found",
  "count":"0"
}
```


Responses for status 2


```
{
  "status":2,
  "status message":"The Ad-ID is valid but has been excluded.",
  "count":1,
  "adid":"SADC0002000",
  "guid":"98786cab"
}
```


Access Denied

```
{
  "status":2,
  "status message": "The Ad-ID has been voided",
  "count":1,
  "adid":"ADID0002000",
  "guid":"492e56f6"
}
```


Voided Code – A user removed this code from the list of active Ad-ID codes.

Data Request Examples (HTML)

Responses for status 0

Ad-ID Found – Full HTML

Ad-ID

Slate

Full Ad-ID code:	ZADE0001000H
GUID:	b1a1dfe
Media Type:	Video
Video Format Flag:	H
Parent:	AD EYE DEE CORP
Advertiser:	AD EYE DEE STORES
Brand:	EYEGLASSES
Product:	REGULAR VISION
Ad Title:	Seeing is Believing
Date Created:	2015-09-25
Copyright:	2015 Ad Eye Dee Corp
Version:	Free case
Agency Name:	Ad-ID, LLC
Language:	English
Length:	30
Bleed:	
Color Type	
Expandable	

Categorization

Industry Group	RETAIL
Major Category	RETAIL STORES
Sub Category	OPTICAL GOODS AND SERVICES
Product Category	OPTICAL CTR

Commercial Delivery

Group	Extreme Reach

Ad-ID Found – Snippet HTML

Slate

Full Ad-ID code:	ZADE0001000H
GUID:	b1a1dfe
Media Type:	Video
Video Format Flag:	H
Parent:	AD EYE DEE CORP
Advertiser:	AD EYE DEE STORES
Brand:	EYEGLASSES
Product:	REGULAR VISION
Ad Title:	Seeing is Believing
Date Created:	2015-09-25
Copyright:	2015 Ad Eye Dee Corp
Version:	Free case
Agency Name:	Ad-ID, LLC
Language:	English
Length:	30
Bleed:	
Color Type	
Expandable	

Categorization

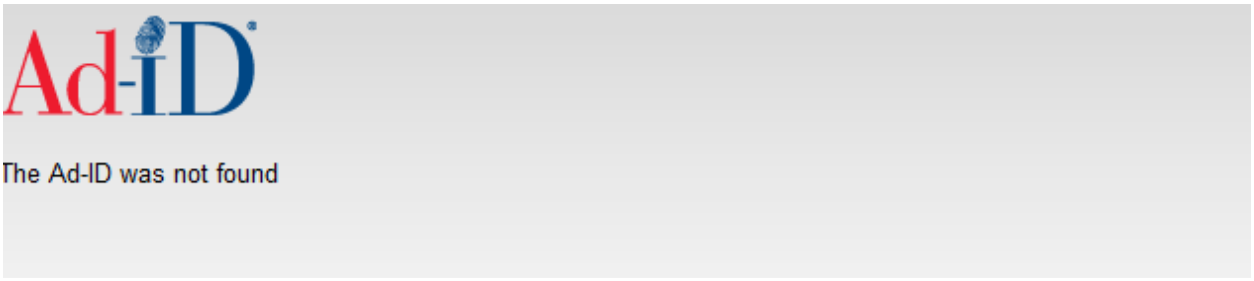
Industry Group	RETAIL
Major Category	RETAIL STORES
Sub Category	OPTICAL GOODS AND SERVICES
Product Category	OPTICAL CTR

Commercial Delivery

Group	Extreme Reach

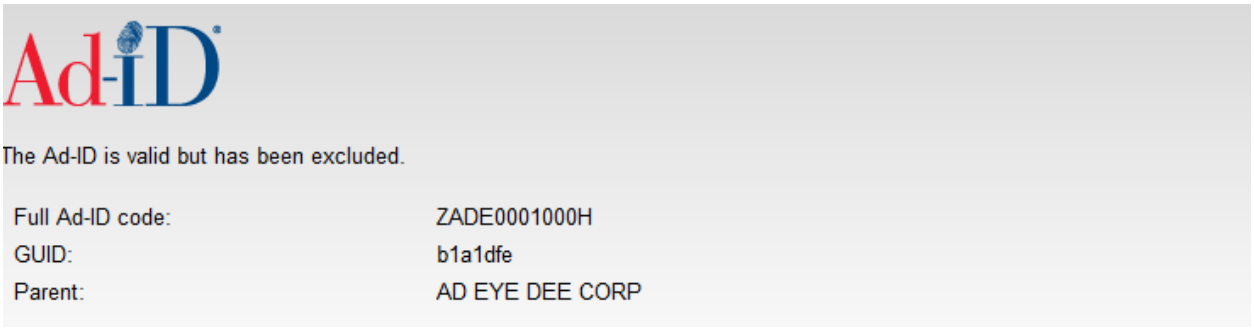
Response for status 1

Ad-ID NOT Found



Responses for status 2

Use Case 1: Access Denied



Use Case 2: Voided Code – A user removed this code from the list of active Ad-ID codes.

Validation Request Examples (XML)

The following are examples of an XML response to a validation request. (Note: HTML responses are not defined for validation requests.)

Response status 0

```
<?xml version="1.0" encoding="UTF-8"?>
 <adids>
  <status>0</status>
  <status_message>The Ad-ID is valid.</status_message>
  <count>1</count>
  <adid>
    <adid_fullcode>501U0015000</adid_fullcode>
    <guid>201984d9</guid>
    <parent>ANYTHING CATS INC</parent>
  </adid>
</adids>
```

Response status 1

```
<?xml version="1.0" encoding="UTF-8"?>
  <adids>
  <status>1</status>
  <status_message>The Ad-ID was not found.</status_message>
  <count>0</count>
 </adids>
```

Validation Request Examples (JSON)

The following are examples of a JSON response to a validation request. (Note: HTML responses are not defined for validation requests.)

Response status 0

```
{
   "status":0,
   "status message":"The Ad-ID is valid",
   "count":1,
   "adid":"SAPC0005000H",
   "guid":"8cddcca",
   "parent":"SODA POP CO"
}
```

Response status 1

```
{
    "status":1,
    "message":"The Ad-ID was not found",
    "count":"0"
}
```

Error Code Reference

The following conditions may generate system actions and error responses as described below:

- Authentication: bad credentials, user ID/API key mismatch, request received outside time limit based on timestamp in message. After X failures in X minutes system will begin to block by IP address. For example, after 5 failures in 5 minutes.
- Malformed request - treated the same as authentication failure.
- Request for nonexistent resource (Ad-ID or cuid not found). After X failures in X minutes, block by IP address for Y minutes. For example, after 5 failures in 5 minutes, block IP 55.55.55.55 for 30 mins.

Errors

Error Code	System Name	Description
1001	CEA_ERR_DATA	There is an error in one or more input data parameters in the request. Malformed Request
2001	CEA_ERR_AUTH	There is an error in the CEA credentials provided in the request or the HMAC is not valid.
5001	CEA_ERR_PROC	A processing error occurred.

System Responses

HTTP Response Code	HTTP Reponses Text	Ad-ID Web Services Usage
200	OK	Success.
400	Bad request	Errors in input data parameters.
403	Forbidden.	Authentication errors or attempting to access data for which the user is not authorized.
500	Internal server error.	An error not covered by the other response codes.

Sample Error Response – XML

```
<?xml version="1.0"?>
<error>
</error_message>
 <error_code>2001</error_code>
 <error_message>Request data failed validation. Please address the issues and try
again.</error_message>
</error>
```

PHP Sample Code for making a CEA request

```
<?php
/**
 * CEA client example script.
 * Usage: php client_example.php <type> <resource>
 * E.g. php client_example.php adid ABCD1234567
 *      php client_example.php cuid abcdefab
 */

// An assigned CEA user id
$userid = 'XXXXXXXX';

// An assigned CEA API key
$api_key = 'XXXXXXXXXXXXXXXX';

$host = 'cea.ad-id.org';




// The CEA service uri. (This example shows a data request.
// For a validation request, subsititute ea_v for ea_c in the
// second component of the uri.
$uri = "/adid_services/ea_c/$argv[1]/$argv[2]";

// An ISO 8601 timestamp. See https://en.wikipedia.org/wiki/ISO_8601
$iso_ts = date("c");

// Construct a signature to send to CEA in an http header.
// See https://tools.ietf.org/html/rfc2104
// and e.g. http://www.freeformatter.com/hmac-generator.html
//
// Per CEA specification, the string to be hashed is constructed
// from the URI and ISO timestamps, assigned above, connected by a
// plus (+) character.
$hash = hash_hmac("SHA256", $uri . '+' . $iso_ts, $api_key);

// The following parameters must be sent in your request to CEA
// as http header fields.
$headers = array();
$headers[] = 'x-userid: ' . $userid;
$headers[] = 'x-date: ' . $iso_ts;
$headers[] = 'x-hash: ' . $hash;

$ch = curl_init();
$timeout = 15;
curl_setopt($ch, CURLOPT_URL, "https://" . $host . $uri);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
```

```
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

// The following cURL options are to ensure secure communication with
CEA.

// Tell cURL to verify the server's certificate
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, TRUE);

// Tell cURL to check that a common name exists in the certificate
// and that it matches the server.
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);

// Point cURL to a certificate for an authority it should trust,
// and by extension whose certificates for server/peer it should
trust.
// (See
http://curl.haxx.se/docs/httpscripting.html#HTTPS_is_HTTP_secure)
// To obtain the authoritative certificate for CEA, browse to cea.ad-
id.org
// with Firefox and follow the instructions here:
// https://support.mozilla.org/en-US/kb/secure-website-certificate
// Place the certificate somewhere that your application can find it.
// In this example, the certificate is located in the same directory
with
// this script.
curl_setopt($ch, CURLOPT_CAINFO, getcwd() .
'/GoDaddyRootCertificateAuthority-G2.crt');

// This option setting is for diagnostic purposes, and may be omitted
curl_setopt($ch, CURLINFO_HEADER_OUT, TRUE);

// Send the request.
$server_output = curl_exec($ch);

// Print diagnostic information, if desired.
print_r(curl_getinfo($ch));

// Close the connection to CEA
curl_close($ch);

// Display the CEA response.
print $server_output;

?>
```