Alexa Web Search Platform Service

Developer Guide

Alexa Web Search Platform Service: Developer Guide

Published 2005-December-08

Table of Contents

Programming Guide

This programming guide is an introduction to using the Alexa Web Search Platform service - what it is, how to access services and how to interpret the output that is returned to you.

To register with AWS, If you have not already registered as an Amazon Web Services developer and received your Access Key ID, point your browser at http://aws.amazon.com and then click the `Create an Account` link.

> **Note**
> All requests to the Alexa Web Search Platform service must be signed. Please see AWS Request Authentication for more information on how to sign requests.

- Making REST requests to the Alexa Web Search Platform service
- Making SOAP requests to the Alexa Web Search Platform service
- AWS Request Authentication

Making REST Requests

This section explains how to use REST (Representational State Transfer) to make requests through the Alexa Web Search Platform service. REST is a Web services protocol that was created by Roy Fielding in his Ph.D. thesis (see Architectural Styles and the Design of Network-based Software Architectures for more details about REST).

REST allows you to make calls to the Alexa Web Search Platform service by passing parameter keys and values in a URL (Uniform Resource Locator). The Alexa Web Search Platform service returns its response in XML (Extensible Markup Language) format. You can experiment with Alexa Web Search Platform service requests and responses using nothing more than a Web browser that is capable of displaying XML documents. Simply enter the REST URL into the browser's address bar, and the browser displays the raw XML response.

The Base URL

Every REST request to the Alexa Web Search Platform service begins with a base URL that is specific to the locale in which you want to make the request. The following base URLs are available:

Access the Alexa Web Search Platform Service

```
http://awis.amazonaws.com/onca/xml?Service=AlexaWebSearchPlatform
```

Request Parameters

The base URL is followed by a series of parameters that define the request. Parameters are separated from the base URL and each other by an ampersand (**`&`**) character. Each parameter consists of a key and a value, separated from each other by an equals sign (**`=`**). Note that parameters and their values are case-sensitive; for example, **`Operation=WebSearch`** works correctly, but **`operation=websearch`** produces an error.

The following example shows a simple REST request that returns a list of results for a custom index specifying currency on a site, where the site's currency is US dollars:

```
http://awis.amazonaws.com/onca/xml?Service=AlexaWebSearchPlatform
    &AWSAccessKeyId=[your Access Key ID here]
    &Operation=WebSearch
    &ResponseGroup=Results
    &Query=currency:USD
    &PublicationId=[PublicationId of a published service that provides your
'currency' index]
    &Timestamp=[timestamp used in Signature]
    &Signature=[Signature calculated from request]
```

The parameters in the example are described in the following table:

Note
To obtain an Access Key ID, you must sign up for an Amazon Web Services Account. When

you sign-up for an Amazon Web Services account, an Access Key ID is generated and associated with your account. To sign-up for an AWS account, please use the sign in page.

Making SOAP Requests

SOAP (Simple Object Access Protocol) lets third-party developers use the Alexa Web Search Platform service by making remote procedure calls. This information is encoded using XML (Extensible Markup Language), although this is transparent to the developer in the course of normal usage. AWIS publishes a Web Services Description Language (WSDL) document that defines all the available Alexa Web Search Platform service APIs, their parameters, and the data that they return (see WSDL Location for more information about the AWIS WSDL).

The SOAP End Point

For Alexa Web Search Platform service data

```
http://awis.amazonaws.com/onca/soap?Service=AlexaWebSearchPlatform
```

AWS Request Authentication

Request authentication is the process of verifying the identity of the sender of a request. In the context of Amazon Web Services (AWS) requests, authentication is the process by which AWS can confirm that a request came from a registered user, as well as the identity of that registered user.

To enable authentication, each request must carry information about the identity of the request sender. The request must also contain additional information that AWS can use to verify that the request can only have been produced by the sender identified. If the request passes this verification test it is determined to be “authentic” and AWS has sufficient information to verify the identity of the sender.

Verifying the identity of the sender of a request is important, as it ensures that only those requests made by the person or party responsible for the AWS account specified in the request are accepted and allowed to interact with AWS services. In this manner, request authentication allows Amazon to track the usage of AWS services on a per request basis. This enables Amazon to charge and bill AWS subscribers for use of AWS paid (not free) services.

AWS Accounts

To access Amazon web services, a developer must create an AWS account. AWS accounts are associated with Amazon.com accounts. To sign in to an AWS account, a developer uses his or her Amazon.com account e-mail and password.

Upon creating the AWS account, the developer is assigned an Access Key ID (*`AWSAccessKeyId`*) and a Secret Access Key. The Access Key ID, which is associated with the AWS account, is used in requests to identify the party responsible for the request. However, because an Access Key ID is sent as a request parameter, it is not secret and could be used by anyone sending a request to AWS. To protect from impersonation, the request sender must provide additional information that can be used to verify the sender’s identity and ensure that the request is legitimate. This additional information, a request signature that is calculated using the Secret Access Key, demonstrates possession of a shared secret known only to AWS and the sender of the request. A Secret Access Key is a 20-character alphanumeric sequence generated by AWS.

Types of AWS Requests

There are two types of requests to AWS:

- *`Anonymous requests`*. Requests to free services can be made anonymously. Though a valid Access Key ID must be included in all requests to AWS, no attempt is made to confirm that the request originated from the party responsible for the AWS account associated with the Access Key ID. In other words, no authentication is necessary for the request to succeed.

  Services to which anonymous requests can be made include: Amazon E-Commerce Service (ECS) 4.0 and Amazon Simple Queue Service (SQS).
- *`Authenticated requests`*. Services that want to track service usage, either for the purpose of calculating usage statistics or to bill for usage, must be able to verify that the identity of the sender of a request is the person or party responsible for the AWS account. The identity of the sender is verified by confirming that the Secret Access Key used in the request signature is the Secret Access Key associated with the Access Key ID included in the request.

  Services to which authenticated requests must be made include: Alexa Web Information Service (AWIS), Alexa Web Search Platform.

To use a service that requires authenticated requests, a signature for each request must be calculated and included as the value of the *`Signature`* parameter in requests to those services.

Authenticating Requests

Requests to AWS are authenticated by verifying information contained within the request. This verification is performed using the following information:

Parameter	Description
`AWSAccessKeyId`	The sender's AWS account is identified by the Access Key ID. The Access Key ID is used to look up the Secret Access Key.
`Signature`	Each request to a web service that requires authenticated requests must contain a valid request signature, or the request is rejected. A request signature is calculated using the Secret Access Key assigned to the developer's account by AWS, which is a shared secret known only to AWS and the developer.
`Timestamp`	The date and time the request was created, represented as a string in UTC. The format of the value of this parameter must match the format of the XML Schema dateTime data type.

Summary of AWS Request Authentication

The following steps are the basic steps used in authenticating requests to AWS. It is assumed that the developer has already registered with AWS and received an Access Key ID and Secret Access Key.

1. The sender constructs a request to AWS.
2. The sender calculates a Keyed-Hashing for Message Authentication code (HMAC), the request signature using the his or her Secret Access Key and the values of the *`Service`*, *`Operation`*, and *`Timestamp`* parameters as input.
3. The sender of the request sends the request data, the signature, and Access Key ID (the key-identifier of the Secret Access Key used) to AWS.
4. AWS uses the Access Key ID to look up the Secret Access Key.
5. AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.
6. If the signature generated by AWS matches the one sent in the request, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Calculating Request Signatures

A request signature, an HMAC, is calculated by concatenating the values of the *`Service`*, *`Operation`*, and *`Timestamp`* parameters, in that order, and then calculating an RFC 2104-compliant HMAC, using the Secret Access Key as the "key." The computed HMAC value should be base64 encoded, and is passed as the value of the *`Signature`* request parameter. For more information, please see http://www.faqs.org/rfcs/rfc2104.html.

When a request is received, AWS verifies the request signature by computing an HMAC value for the request and comparing the value of that HMAC with the value in the request. If the computed HMAC value matches the HMAC value in the request, the identity of the sender is verified and the request is accepted. If the values do not match the request is rejected, and an error is returned.

Using REST and SOAP Transactions

Requests can be sent using REST (XML over HTTP) or SOAP. The contents of the request are the same, only the request format differs.

URL Encoding

The result of the SHA-1 hash is binary data. An encoding must be specified to include this in either a SOAP or REST request. Both REST and SOAP requests should be Base64 encoded.

However, as the results of Base64 encoding can contain characters that are not legal in a URL, such as plus signs (+),slashes (/), and equal signs (=), results for REST requests should be URL encoded, as specified in RFC 1738, section 2.2.

Code Samples for Request Authentication

Calculating an HMAC Request Signature

The following code sample demonstrates how to calculate a request signature to sign authenticated requests to AWS.

```
package amazon.webservices.common;

import java.security.SignatureException;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

/**
 * This class defines common routines for generating
 * authentication signatures for AWS Platform requests.
 */

public class Signature {

    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     *
     * @param data
     *     The data to be signed.
     * @param key
     *     The signing key.
     * @return
     *     The Base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     *     java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
        throws java.security.SignatureException
    {
        String result;
        try {
            // get an hmac_sha1 key from the raw key bytes
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM);

            // get an hmac_sha1 Mac instance and initialize with the signing
key
            Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
            mac.init(signingKey);
            // compute the hmac on input data bytes
            byte[] rawHmac = mac.doFinal(data.getBytes());

            // base64-encode the hmac
            result = Encoding.EncodeBase64(rawHmac);
        }
        catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " +
e.getMessage());
        }
        return result;
    }
}
```

Data Encoding

This sample, provided in support of the previous sample for calculating HMAC signatures, demonstrates how to perform Base64 encoding of input types in AWS requests.

```
package amazon.webservices.common;

/**
 * This class defines common routines for encoding
 * data in AWS Platform requests.
 */

public class Encoding {

    /**
     * Performs base64-encoding of input bytes.
     *
     * @param rawData
     *      Array of bytes to be encoded.
     * @return
     *      The base64-encoded string representation of rawData.
     */
    public static String EncodeBase64(byte[] rawData) {
        return Base64.encodeBytes(rawData);
    }
}
```

Performing Base64 Encoding and Decoding

This sample demonstrates how to encode and decode to and from Base64 notation. The code for this sample is not included in this document due to the length of the file. The code, which is public domain, can be accessed using this link: http://iharder.net/base64.

Alexa Web Search Platform Service API Reference

Introduction

This section contains details about the Alexa Web Information Service, including the Operations, Response Groups, and other elements that make up the application programming interface (API).

Each Operation listed contains at least one sample request to help you get started. Use the sample requests as a starting point for developing your own requests. Keep in mind that you should substitute your own Access Key ID (`AWSAccessKeyId`) into the requests before using them.

Contents

WSDL and Schema Locations

The Alexa Web Search Platform service publishes its API through Web Services Description Language (WSDL) documents, which you can use to construct SOAP requests. XML schemas are also provided for validating the XML output of REST requests. WSDL documents and XML schemas are available for different versions and different locales.

Default WSDL and Schema locations

Service	URL
Alexa Web Search Platform	`ht-tp://awis.amazonaws.com/AlexaWebSearchPlatform/AlexaWebSearchPlatform.wsdl`

The following table shows the location of the latest XML schema:

Service	URL
Alexa Web Search Platform	`ht-tp://awis.amazonaws.com/AlexaWebSearchPlatform/2005-12-01/AlexaWebSearchPlatform.xsd`

Operations

The Alexa Web Search Platform service currently contains one operation, WebSearch, that returns results from the Alexa search engine. Queries can be customized with new search fields that are created on the Alexa Web Search Platform.

Operations

- `WebSearch`

WebSearch Operation

Description

The Web Search Operation may be used to retrieve a list of search results that match a query. Users can include their own custom search fields in the Alexa Search.

Sample Request

Using Web Search Operation

The following `WebSearch` example demonstrates how to make a REST request.

```
http://awis.amazonaws.com/onca/xml?Service=AlexaWebSearchPlatform
&Operation=WebSearch
&AWSAccessKeyId=[Your AWS Access Key ID]
&Signature=[signature]
&Timestamp=[timestamp used in signature]
&ResponseGroup=[Valid Response Group]
&Query=[Search terms]
&PublicationId=[Publication ID]
&TimeOut=[Time in milliseconds]
&Unique=[Site,2 | Shingle,2/6 | Site,2;Shingle,2/6]
&Relevance=[number between 0 and 4]
&SearchFields=[Lists of search field names.]
&Start=[number to start | -1 (for count-only)]
&Count=[maximum number of results]
```

For more information on making signed requests, see Signing Requests

Request Parameters

The WebSearch Operation takes the following parameters. Required parameters must be provided for the request to succeed.

Name	Description	Type	Value
`Operation`	Use the *`Operation`* parameter to specify the name of the operation you would like to call. To access the `Web-Search` operation, set the *`Operation`* para-	Required	*`WebSearch`*

Name	Description	Type	Value
	meter to **`WebSearch`**.		
`ResponseGroup`	Any valid response group. See the Response Group section for valid options.	Required	Comma-separated list of response groups.
`Query`	A set of search terms. Phrases must be encapsulated in single quotes. Note, all requests must be escaped.	Required	Search terms
`PublicationId`	The publication id tells the service which user-created search indices to use. This id is provided by the Alexa Web Search Platform portal (http://websearch.alexa.com) when you create a new search service.	Required	Publication ID
`Version`		Optional	
`TimeOut`	Time in milliseconds to wait for a response. Web Search will return as many results as possible within the timeout period. Default value is '3000.' Maximum value is '9000.'	Optional	Time in milliseconds
`Unique`	Specify unique options. Multiple options can be delimited with a ';'. Default value is 'Site,2;Shingle,2/6;siteprefix.subsite.site.path'.	Optional	Site,2 | Shingle,2/6 | Site,2;Shingle,2/6
`Relevance`	Allows faster searching with weak relevance (0) to slow searching with strong relevance checking (4). Non-zero values also require the use of the 'SearchFields' parameter (which provides a default). Default value is '4.'	Optional	number between 0 and 4

Name	Description	Type	Value
SearchFields	The set(s) of fields to search for Relevance. Required when 'Relevance' parameter is non-zero. If there are multiple sets of fields(separated by ';'), each set is tried in turn until the specified count is reached (see 'Count' parameter). Within a set, multiple fields are separated by ','. Default value is 'Site,Url,Title,Dmoz,Keywords,SLD;Site,Url,Title,Dmoz,Keywords,SLD,Text'.	Optional	Lists of search field names.
Start	Number of result at which to start. Used for paging through results. Default value is '0.' One can request a 'count-only' response by specifying a start value of '-1.' Note that when requesting count-only, use of 'Unique' parameter is highly discouraged and such requests will likely timeout.	Optional	number to start | -1 (for count-only)
Count	Number of results (maximum) per page to return. Note that the response document may contain fewer results than this maximum. Default value is '10' (maximum 20).	Optional	maximum number of results

Response Groups

Response groups allow the user more control over what data is returned. By specifying one or more response groups when making the request, you can retrieve only the information you are interested in.

Response Group	Description		
Results	Search the Alexa web search service for spe-		

Response Group	Description		
	cified keywords.		
Context	Search the Alexa web search service for specified keywords and return the results with the context. This operation is slower than searches without context.		

Advanced Query Syntax for the WebSearch Operation

General Query Syntax

The Query parameter specifies the criteria used to search the Alexa web archive.

Operators

Type	Example	Description
"AND"	`cat dog`	Words are ANDed together by default. Search for documents that contain the word cat and the word dog

"NOT"

Use "-" to mean NOT. Search for documents that contain the word cat, but not the word dog:

```
cat -dog
```

Phrases

Use double quotes (") to make phrases. Search for for documents that contain the phrase "cats and dogs":

```
"cats and dogs"
```

"OR"

Use "|" to do OR. Search for documents containing the word "cat" or the word "dog":

```
cat | dog
```

Grouping

Use parentheses to group terms. You can nest groups as well, e.g. "((a | b) c)". Search for documents containing the the word "cat" or the word "dog", and the word "control":

```
(cat | dog) control
```

Wildcards

Use "*" for a placeholder in phrases, e.g. "a * c" matches "a b c" or "a x c" but not "a c" or "a b b c". Search for documents that contain phrases matching "The cat *[some word]* on the bed":

```
the cat * on the bed
```

Available Search Fields

You may also specify individual search fields to query by prefixing words with the field name. The complete list of search fields is shown below. The fields allow you to search for documents based on attributes of the document, on the website the document is hosted on, the URL of the document, or pages redirecting to documents, or linking to documents.

Examples:

Search for all JPEG images on yahoo.com:

```
Type:image/jpeg Site:Yahoo.com
```

Search for pages containing the words "cat" and "dog" in the title that are in the French language:

```
Title:(cat dog) Lang:fr
```

Type	Field Name	Description
Document	Text	Text of document, sans markup
	Code	HTTP response code returned by server at crawl time
	Title	Document title
	Type	MIME type from header (text/plain, image/jpeg, jpeg, . . .)
	Lang	Two character language code (en, fr, ja, . . .)
	Charset	Character set (utf-8, big5, iso-8859, . . .)
	LinkText	Outbound anchor text
	Porn	Document contains adult content (yes, no, maybe)
	SizeAtLeast	Minimum document size (512, 1k, 2k, 4k, . . . 256k, 512k, 1m, 2m)
	Magic	MIME type from content (text/plain, image/jpeg, jpeg, . . .)

Type	Field Name	Description
Website	Traffic	Alexa traffic rank (Top5, Top10, Top50, . . .Top10000000)
	Dmoz	Dmoz categories (Arts, Business/Accounting, . . .)
URL	Site	Site (members.aol.com/~bob, yahoo.com, . . .)
	Url	URL ("aol.com/login?loc=us" from my.name.aol.com/login?loc=us)
	SubSite	Sub-site ("name" from my.name.aol.com)
	SitePrefix	Site prefix ("my" from my.name.aol.com)
	Domain	Domain ("amazon.co.uk" & "co.uk" & "uk" from www.amazon.co.uk/path?)
	SLD	Second level domain ("amazon" from www.amazon.co.uk/path?)
	Suffix	URL suffix ("doc" from aol.com/test.cgi?foo=bar.doc)
	CSuffix	Pre-query suffix ("cgi" from aol.com/test.cgi?foo=bar.doc)
Redirecting to	RSite	Site (members.aol.com/ bob, yahoo.com, . . .)
	RUrl	URL ("aol.com/login?loc=us" from my.name.aol.com/login?loc=us)
	RSubSite	Sub-site ("name" from my.name.aol.com)
	RSitePrefix	Site prefix ("my" from my.name.aol.com)
	RDomain	Domain ("amazon.co.uk" & "co.uk" & "uk" from www.amazon.co.uk/path?)
	RSLD	Second level domain ("amazon" from www.amazon.co.uk/path?)
	RSuffix	URL suffix ("doc" from aol.com/test.cgi?foo=bar.doc)
	RCSuffix	Pre-query suffix ("cgi" from aol.com/test.cgi?foo=bar.doc)
Linking to	LSite	Linking to Site (members.aol.com/ bob, ya-

Type	Field Name	Description
		hoo.com, . . .)
	LUrl	URL ("aol.com/login?loc=us" from my.name.aol.com/login?loc=us)
	LSubSite	Sub-site ("name" from my.name.aol.com)
	LSitePrefix	Site prefix ("my" from my.name.aol.com)
	LDomain	Domain ("amazon.co.uk" & "co.uk" & "uk" from www.amazon.co.uk/path?)
	LSLD	Second level domain ("amazon" from www.amazon.co.uk/path?)
	LSuffix	URL suffix ("doc" from aol.com/test.cgi?foo=bar.doc)
	LCSuffix	Pre-query suffix ("cgi" from aol.com/test.cgi?foo=bar.doc)