System1 Embedded Search SDK

Segments

A segment is the identifier used in embedded-search to identify a bucket of your traffic. This key is used to provide the correct configuration of results on your pages, as well as to identify your traffic in our reporting system. Work with your partnership manager to configure your segments.

Regions

The embedded search service is built around the “regions” on your page. In HTML notation, these are the container s that the javascript will place the content like ads, organic results, and related searches.

A very simple search page will have exactly one content div, typically called the “mainline” div. It’s also possible to have a “sidebar” div where things like related searches will be displayed.

For some implementations, you only want ads and you want them to be placed above and below your mainline search results. We typically call those content divs “mainline-top” and “mainline-bottom”. You can still have a “sidebar” content div in this case, too.

These are just guidelines - there’s no set requirement of what the divs are named and where they are located on the page, as long as you communicate it clearly with your account manager and stay consistent on a per-segment basis.

Implementation Steps

Communicate Requirements to your Account Manager

The first step is to share details of your proposed implementation with your account manager. The key things to include are:

1. The fully qualified domain name that will run the embedded search javascript (e.g. www.dogpile.com)
2. The regions on your SERP (see the next section)
3. The combination of ads and/or organic results you want to see rendered on the page.

Get Parameters from your Account Manager

After communicating all the requirements to your Account Manager, they should give you back the following things:

1. Your partner ID.
2. Your access key (make sure to keep this private!)
3. Your segment name(s).

4. Your region names.

JavaScript Changes

Now you're ready to finish the integration. After your web server has calculated the signature token using the access key (see the section below on the Authentication Algorithm), the web page returned to the browser client will include a few lines of JavaScript required to request and render search results.

Include reference to System1 JS

In your web application include a reference to System1's client-side search results JS at the end of the body tag, the JS includes configuration settings specific to your segment.

```
<script async type="text/javascript"
src="https://s.flocdn.com/@s1/embedded-search/embedded-search.js">
</script>
```

Construct page elements

The System1 JS will execute an asynchronous client-side call to System1 for search results, and the JSON response will be used by the System1 JS to build the page markup that makes up the Search modules. All search results will render simultaneously on the page. For each search results container include a on the page where it should appear. For each include the `data-s1search` attribute that matches the containers specified to you by your account manager. Sample HTML:

```
<div data-s1search="mainline-top"></div>
<div data-s1search="mainline-bottom"></div>
<div data-s1search="sidebar"></div>
```

Configure s1search() request

The parameters you include in the s1search() call will be passed to System1's search service to retrieve search results. Include the script block to call s1search(), as seen in the below example, in the <head> tag of your page.

Sample s1search() call and parameters

```
<script>
        function onResolved() {
            console.log('completed');
        }

        function onRejected(msg) {
```

```
            console.error(msg);
        }

        window.s1search = window.s1search || function () {
(window.s1search.q = window.s1search.q || []).push(arguments) };

        window.s1search('config', {
            category: "web",
            domain: "{{ your domain }}",
            partnerId: {{ your partner id }},
            isTest: true,
            onComplete: onResolved,
            onError: onRejected,
            query: "{{ query }}",
            segment: "{{ segment }}",
            signature: "{{ csr signature method }}"
        });
    </script>
```

Key parameters:

Key	Description	Accepted Values	Default
gdprOptIn	This parameter specifies whether the publisher has obtained consent from users from EU countries, as specified by GDPR (https://www.eugdpr.org) for processing of his or her personal data, including writing cookies and personalization tracking. • Set to false, it means the publisher has not obtained consent per GDPR requirements • Set to true, it means the publisher has obtained consent per GDPR requirements Note: This parameter only affects users originating in EU countries	Boolean, true or false	Optional. Default is false in EU countries. (This parameter is ignored in non-EU countries.)

	as determined by IP address/geolocation.		
domain	The fully qualified domain that this implementation is embedded in.	String	Required
partnerId	A unique identifier for you; mainly related to billing purposes.	Integer	Required
query	The user's search query. Ensure that this is the same exact query term that was used to generate the signature value below.	String	Required
querystringParams	Where the SDK should look for the page and query string parameters in the URL. For example, if "page" is set to "p" and "query" is set to "s" then the url http://www.dogpile.com/?s=dogs&p=2 means the query is for "dogs" and the page number is two. Whatever is not specified gets the default value.	Object with "page" and "query" attributes	{"page": "page", "query": "q"}
segment	Work with your Partnership Manager to obtain your segment.	String	Required
signature	Encoded hashed value of timestamp, access key, and query terms.	String	Required
category	Sets the category for the user's search.	web, images, video, news, shopping	web
subId	Supplemental subId parameter used in addition to sub params on the source page url. Must be configured; work with your Partnership Manager to enable.	String	Optional
clickTrackingUrl	This is the URL that will contain the information you want to receive on a end-user click. For	String	Optional

	more information on this topic, refer to this section. Limit the length of the URL to 1000 chars or less.		
isTest	Used to indicate to content providers that the request is a test and should not be included in revenue collection.	Boolean, true or false	false
onComplete	This callback is invoked once the results have been rendered. It will contain additional information about the results that were returned. Further details will be forthcoming.	JavaScript callback	Optional
onError	If there are any errors or warnings that occurred during the call, this callback will be invoked with a list of those errors and warnings.	JavaScript callback	Optional

Note on Testing from Staging & Development Environments

If you have a staging or development environment you want to setup your integration with, you can do so with the following steps:

1. Pass in the `domain` parameter of the production domain. So, let's say you're serving live traffic on `www.system1.com`, and you want to do dev work on `dev.system1.com`. Set the `domain` in the `s1search` call to `www.system1.com`.
2. Set the `isTest` flag to `true`.

Signing the Request

System1's embedded-search API requires you to sign all requests. This allows the API to validate that the requests originated from your application, and prevents unauthorized access to the API using your credentials. Without a valid signature, the requests will be rejected, and no results returned.

Prerequisites:

- A System1 search segment
- A System1 search access key
- The system clock on each server that signs a request to be accurate within a maximum deviation of 1 minute.
- The signature is a specially formed binary hash of the following three values:
    1. Request date and time
    2. Access key/token
    3. Query term(s)

Note: The query term used to generate the signature must be the exact same query term that is passed to the s1search() request. For example, if the query term will be trimmed of trailing spaces before making the request to s1search(), ensure this happens prior to generating the signature.

Algorithm

The basic algorithm for creating the signature starts with creating a timestamp as follows:

1. Start with the UTC/GMT time
2. Round to the nearest minute (30 seconds or greater rounds up, otherwise round down)
3. Format the time as a string: yyyyMMddHHmm
    - yyyy - 4 digit year
    - MM - 2 digit month, 01-12
    - dd - 2 digit day, 01-31
    - HH - 2 digit hour, 24-hour clock, 00-23
    - mm - 2 digit minute, 00-59
4. Concatenate the values together in the following order:
    - timestamp
    - access key
    - query term(s) (if you have no query terms use the empty string)
5. Encode concatenated string into a UTF-8 byte encoding.
6. Perform a SHA-256 (preferred) or SHA-1 hash of the UTF-8 encoded string value.
7. Encode the binary hashed value using url-safe base-64 encoding and trim any padding characters (“=“ sign in base64); see http://tools.ietf.org/html/rfc4648#section-5
    - Caution: some languages’ hash functions default to a hex output instead of a binary output (e.g. PHP is a common language that does this); ensure you are base64 encoding the **binary** hash output, not the hex representation.
    - Note: url-safe base64 encoding is a variation on standard base64 encoding and results in slightly different output. Consult your language documentation for an implementation or review the link above.

Example implementations in several programming languages can be found here for reference: http://www.infospace.com/partners/sdk/csr/signingSample.html.

Below is a step by step sample of the outputs during signature construction that can be used to help troubleshoot any issues you may be having.

Starting with the following sample inputs:

Time: `January 3rd, 2020 13:28:56`
Token: `VKxZxMDp_hx2tchU1M6eWRfn`
Query Term: `ipad`

1. Start with the UTC/GMT time -- `January 3rd, 2020 13:28:56`
2. Round to the nearest minute -- `January 3rd, 2020 13:29:00`
3. Format the time as a string -- `202001031329`
4. Concatenate the values together -- `202001031329VKxZxMDp_hx2tchU1M6eWRfnipad`
5. Encode concatenated string into a UTF-8 byte encoding -- `202001031329VKxZxMDp_hx2tchU1M6eWRfnipad`
6. Perform a SHA-256 (preferred) or SHA-1 hash of the UTF-8 encoded string value. **NOTE:** This string representation of the hashed value may look different depending on what coding language you're using to compute the hash. The important thing (also noted above) is to make sure that you're using the binary hash output and NOT the hex representation when moving onto step 7 -- `\x1d\x86\xca\x17\x14,\x1dY\x1f"\xf7\xe7\xd6}k\xf4G\xb5\xa07pd\xa8\x8eM_UoVR\xb7\xaa`
7. Encode the binary hashed value using url-safe base-64 encoding and trim any padding characters -- `HYbKFxQsHVkfIvfn1n1r9Ee1oDdwZKiOTV9Vb1ZSt6o`

Click Tracking in Search Results

Goal: gain deeper insights to user behavior on an System1 search experience with our Site Tracking solution. System1 enables partners to receive data related to a click event for reporting and analysis.

How it works

The partner provides a click handler URL that the System1 click handler calls every time a click event occurs. This URL may contain custom data via query string parameters that a partner needs with every click, (e.g. Token) that will be passed back every time the URL is called. In addition to any custom data, the System1 click handler can supply a number of variables that characterize that particular click.

Supported parameters:

Key	Description	Type	Sample value

{info[domain]}	The fully qualified domain this click happened on.	String	www.dogpile.com
{info[page]}	The page number that this click happened on (zero indexed).	Integer	0
{info[query]}	The query on the current serp.	String	dogs
{info[query_category]}	The query category (web, images, news, videos, etc.)	String	web
{info[segment]}	The segment the click happened on.	String	info.0001
{info[subsequent_search]}	Whether or not this page is a subsequent search	Boolean	false
{requestu_args[ip]}	The end user's IP address	IP Address	10.23.125.23
{requestu_args[user_id]}	The user's cookie. This cookie is unique to the browser and has a 1 year expiry.	String	FqUyptgETnJXSgqC3gxk
{info[persist_args][gclid]}	The Google Ads click tracking parameter, gclid.	String	….
{info[persist_args][msclkid]}	The Bing Ads click tracking parameter, msclkid.	String	...
{extra_args[p]}	Whether or not this is a paid or non-paid click	Integer	1
{extra_args[b]}	The backend that the ad or algo result comes from.	String	google
{extra_args[position]}	The zero-indexed position of the ad or algo result that was clicked.	Integer	1

{extra_args[advertiser_domain]}	The Bing Ads advertiser domain for a paid click.	String	www.domain.com

To begin, pass the URL in as the clickTrackingUrl parameter in s1search() call. If you are a hosted partner, provide your partnership manager with your click handler URL.

A sample URL used to forward the click to the partner click handler looks like this (The example below is for documentation purposes only, and is not intended to be used):

```
http://partner.clickserver.com/ClickHandler?partnerCustomParameter=val
ue1&secondParameter=value2&page={info[page]}&query={info[query]}&ip={r
equestu_args[ip]}&paid={extra_args[p]}&backend={extra_args[b]&position
={extra_args[position]}
```

Which would call your server as the following:

```
http://partner.clickserver.com/ClickHandler?partnerCustomParameter=val
ue1&secondParameter=value2&page=0&query=dogs+and+cats&ip=139.31.222.12
&paid=1&backend=google&position=2
```