

Connected Search

Table of Contents

connected search

Springboard	3
Springboard UI	6
Applications Manager	7
Applications ..	9
Usage ..	10
Data sources ..	11
Users ..	12
Applications UI	13
Metrics Hub ..	15
Experience Optimizer ..	23
Data Sources ..	29
Integrations ..	35
Applications	36
Users	37
Roles	38
Data sources	41
Web data source	42
Custom metadata tags ..	51
Robots ..	53
Troubleshooting guide ..	54
Field mapping ..	58
Push data source	62
Language support	63
Developer documentation	67
Authentication API	69
Push API	72
Query API	76
Search ..	82
Searchahead ..	87
Detail ..	91
Metatags ..	95
Browse ..	96
Related Content ..	100
HTTP response errors	106
Signals API	113
API credentials	114
Integration tools	115
Embeds	116
Breadcrumbs ..	120
Configuration ..	121

CSS styling ..	122
Facets ..	123
Configuration ..	124
CSS styling ..	127
Related content ..	128
Configuration ..	129
CSS styling ..	131
Searchahead ..	132
Configuration ..	134
CSS styling ..	138
Search results ..	139
Configuration ..	140
CSS styling ..	143
Search Store	144
API functions ..	145
Search ..	149
Searchahead ..	151
Detail ..	153
Browse ..	155
Metatags ..	157
Signals ..	159
Observable and stateful functions ..	162
Signals	165
Regions	166
Relevancy controls	167
Support	170

Springboard

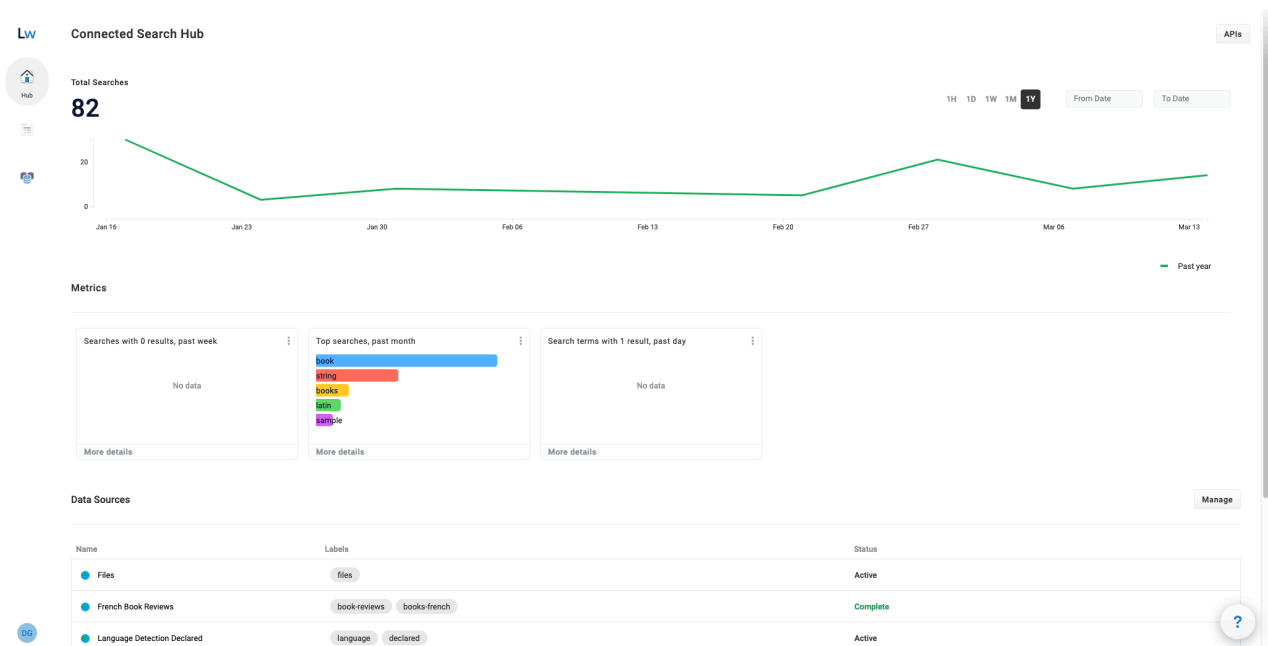
Concept

Springboard powers applications for search and customer service and allows non-technical users to achieve key performance indicators (KPIs). Springboard applications help solve challenging business issues by providing:

- Improved time to value
- Simplified maintenance
- Relevancy controls
- Low risk, low effort search setup and integration

Connected Search

Connected Search is designed for site search. Index and search high-volume sites within minutes with built-in relevancy driven by machine learning.



Capabilities

Data sources

Data sources are complete data ingestion configurations that decide how to handle your data during ingestion by Springboard. When you run a fully configured data source, the result is a search-optimized indexed data set.

Learn more about the available data sources.

APIs

Operations APIs let you work with documents and other resources exposed by Springboard.

The *Query API* is used for various query operations, including getting results from search, searchahead, and other search-related endpoints.

The *Push API* is used to push structured data to Springboard to make it searchable.

The developer documentation offers tools to help you learn more about Springboard APIs. Springboard *authenticates* using OAuth 2.0 client credentials grant type [↗](#), a standard that defines a secure protocol for protecting your data.

API Details

QUERY APIS

Search

POST [https://api.lucidworks.com/query/{customerUuid}/\(applicationUuid\)/search](https://api.lucidworks.com/query/{customerUuid}/(applicationUuid)/search)

Finds relevant matches to complete terms within an indexed data collection. Results match all of the words in the query. For partial matches use the `searchahead` endpoint.

Request

Security: Bearer Auth

For details about creating an authentication token, see the Authentication page. This token expires after one hour. A new token is required at that time.

Provide your bearer token in the Authorization header when making requests to protected resources.

Example: `Authorization: Bearer 123`

Path Parameters

applicationUuid string
The unique ID assigned to an application in a customer environment. The value is displayed in the Springboard application's API modal. required

customerUuid string
The unique ID assigned to a customer environment. The value is displayed in the Springboard application's API modal. required

Auth

Token : 123

Parameters

applicationUuid : string

customerUuid : string

Body

```
{
  "query": "string",
  "utcOffset": "+0400",
  "sessionId": "string",
  "page": 0,
  "limit": 20,
  "sort": {
    "sortField": "score",
    "sortOrder": "desc"
  }
}
```

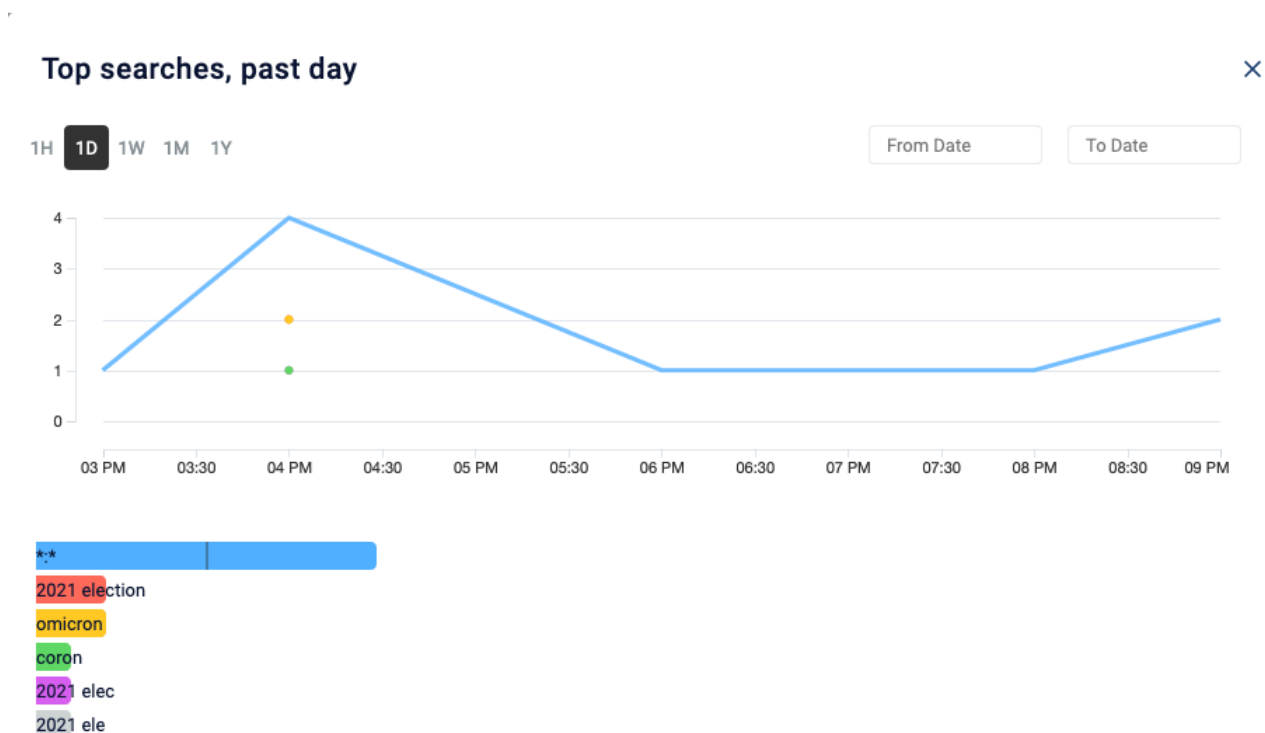
Send API Request Production

Request Sample: Shell / cURL

```
curl -request POST \
  -url https://api.lucidworks.com/query/{customerUuid}/(applicationUuid)/search \
  -header 'Authorization: Bearer ' \
```

For more information, see [Developer documentation](#).




Analytics



Learn more about analytics.

Fine-tuned relevancy

Experience Optimizer is the results configuration area for your application. Experience Optimizer lets you test, visualize, and improve search results so your end users have the best search experience.



Facets

TYPE

Page134

DATASOURCE LABELS

bbc134

DOCUMENT LANGUAGE CODE

en182

es2

fa2

ur2


vi2


Show More


Results List


Showing 1 - 20 of 134 results


Sort byRelevance

**BBC - Homepage**
Breaking news, sport, TV, radio and a whole lot more.
The BBC informs, educates and entertains - wherever you are, whatever your age.
<https://bbc.com/>

**Future Planet - BBC Future**
As we face the world's greatest environmental challenges, what we need most is solutions. Future Planet brings you stories of the ways the world can become a more sustainable place.
<https://www.bbc.com/future/future-planet>

**Yazidi genocide: IS member found guilty in German landmark trial - BBC News**
Taha al-Jumaily is jailed by a German court for crimes including the murder of a young Yazidi girl.
<https://www.bbc.com/news/world-europe-59474616>

**BBFC tightens rules over racist language in films - BBC News**
Any film now featuring the N-word is highly unlikely to be classified lower than a 12A/12.
<https://www.bbc.com/news/entertainment-arts-59473980>

**Tennis - BBC Sport**
The home of Tennis on BBC Sport online. Includes the latest news stories, results, fixtures, video and audio.
<https://www.bbc.co.uk/sport/tennis>

Show: 20 < Previous 1 2 3 ... 6 7 Next > Jump to: 1

Learn more about Experience Optimizer.

Springboard UI

Concept

The topics in this section describe the primary UI elements of Springboard. When you sign in to Springboard, the *Springboard Applications Manager* screen displays.

The upper section of the screen contains a list of your applications. You can select an existing application to view the *Applications UI*, where you can manage the data sources, curate search results, or review analytical data. You can also click **+New** to add a new application.

The lower section of the screen displays a list of your Springboard users and their status. With the appropriate permissions, you can select an existing user to view, edit, or delete the user account. You can also click **+New** to create a new user. For more information, see *Users*.

Applications Manager



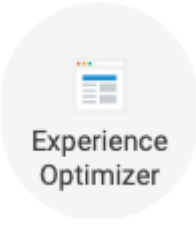


Concept

The Applications Manager is the starting point for managing Springboard and displays your applications and users and their current status.

Sidebar

The sidebar provides quick access to screens within your application, including the Applications Manager, Hub, Experience Optimizer, data sources, and integration settings. Use the sidebar to navigate your application.

The icon for the current page displays with the name below it.

Icon	Description
	Return to the <i>Applications Manager</i> screen.
	Display the application's <i>Hub</i> screen to view a summary of your application.
	Open <i>Experience Optimizer</i> to curate results for data source crawls.
	Open the <i>Data Sources</i> screen to view, edit, delete, or add new URLs to crawl.
	Open <i>Integrations</i> to configure embeds and obtain API credentials.



Click your user profile to sign out of Springboard.

Applications

Applications Manager

Concept

The Applications screen displays all the applications you have created in Springboard. An application is a named set of linked objects, including data sources. Generally, you will create different Springboard applications for different purposes.

An active, configured application does not display a status in the top right corner of the application. Applications that are not active display a status. For example:

- **Ready to Configure.** The application has been created and is ready to connect to a data source.
- **Service Issue.** The application has encountered an issue and is not currently functional.

Additional information

- For conceptual information about active, configured applications, see *Applications UI*.
- For information about managing applications, see:
 - *Create a Springboard application*
 - *Manage Springboard data sources*

Usage

Applications Manager

Concept

The Usage screen displays the requests and activity across all of your Springboard applications. This screen allows you to review your Springboard use according to your contract terms.

Total requests activity

The Total Requests Activity chart displays the total number of requests processed by Springboard per day from all your applications. Each request to the Springboard APIs is included here. These requests are not divided by request type or by application.

Total Requests

12,456

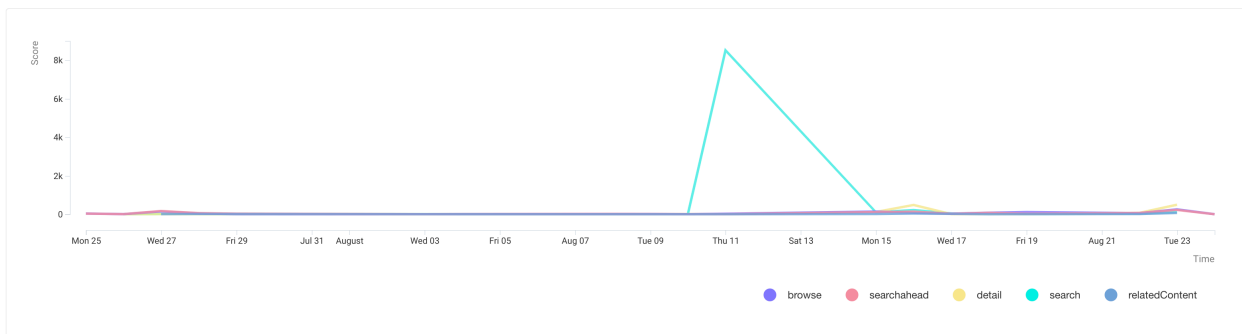
(+0%) Past Month



Use cases

The Use Cases chart displays the total number of requests per day by use case, which correspond to the *API endpoints*.

Use Cases



Note

Visit *an application's Hub* to view the data for that application.

Data sources

Applications Manager

Concept

The Data Sources screen displays all data sources from all of your Springboard applications.

Data sources are complete data ingestion configurations that decide how to handle your data during ingestion by Springboard. When you run a fully configured data source, the result is a search-optimized indexed data set.

Users

Applications Manager

Concept

The Users screen displays all the users associated with your Springboard instance. The name, email, and status of each user display in a table.

Select a user to view their *roles*.

Additional information

- For conceptual information, see *Users*.
- For information about managing users, see:
 - *View users*
 - *Invite users*
 - *Edit or delete users*

Applications UI



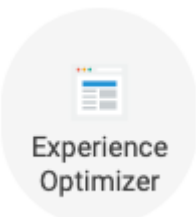


Concept

When you select an application from the *Applications Manager* screen, the application's user interface displays. The elements described in this topic display on multiple screens in Springboard. However, the central portion of this screen is information specific to the *Hub* screen.

Sidebar

The sidebar provides quick access to screens within your application, including the Applications Manager, Hub, Experience Optimizer, data sources, and more. Use the sidebar to navigate your application.

The icon for the current page displays with the name below it.

Icon	Description
	Return to the <i>Applications Manager</i> screen.
	Display the application's <i>Hub</i> screen to view a summary of your application.
	Open <i>Experience Optimizer</i> to curate results for data source crawls.
	Open the <i>Data Sources</i> screen to view, edit, delete, or add new URLs to crawl.
	Open <i>Integrations</i> to configure embeds and obtain API credentials.



Click your user profile to sign out of Springboard.

Additional elements

- Select the support icon to *contact support for assistance* with Springboard.



Metrics Hub

Applications UI

Concept

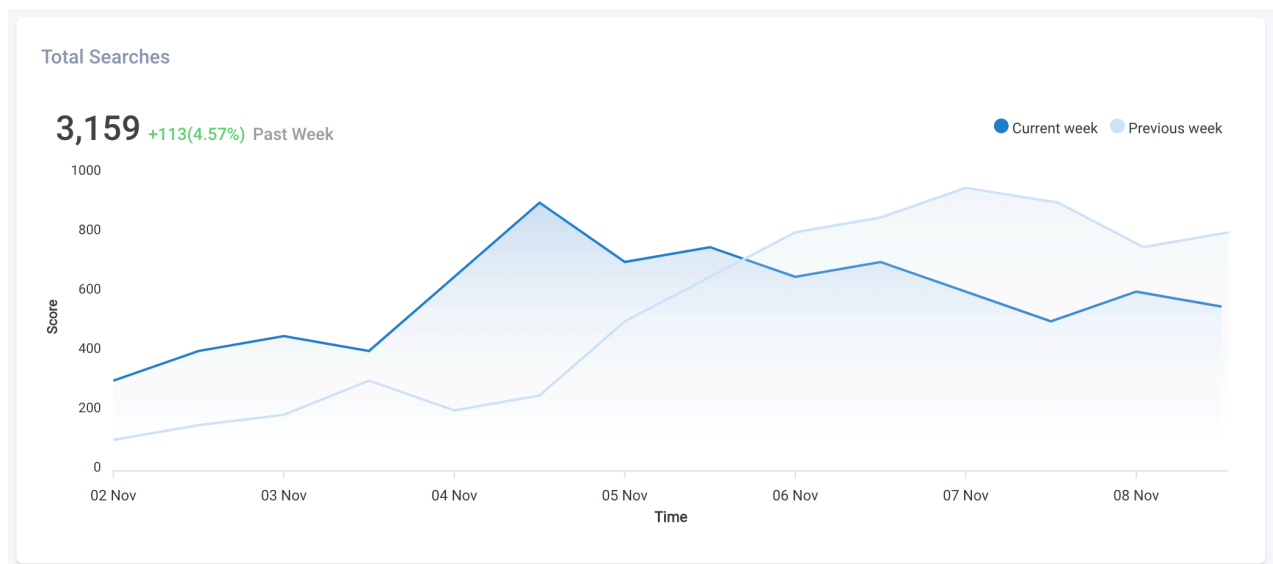
When you select a Springboard application from the *Applications Manager* screen, the application's user interface displays. The elements described in this topic display in the central portion of the screen and are specific to the Metrics Hub screen. The other elements display on multiple screens in Springboard and are detailed in *Applications UI*.

The Metrics Hub displays your application's search activity with a period selector for the past hour, day, week, month, year, or custom interval. You can customize the Metrics Hub by selecting and saving a preset or custom time range.

Information such as null results, top searches, and search terms with one result display in the Metrics Hub.

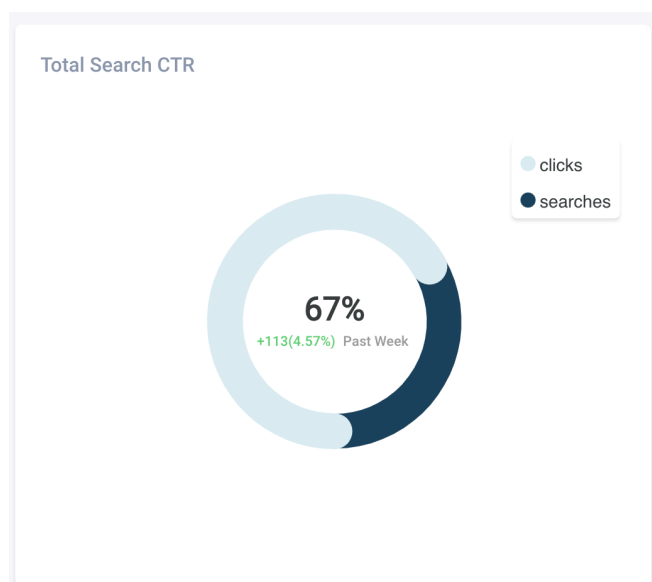
Total searches

The Total Searches graph shows the number of searches over the selected time interval compared to the previous period.



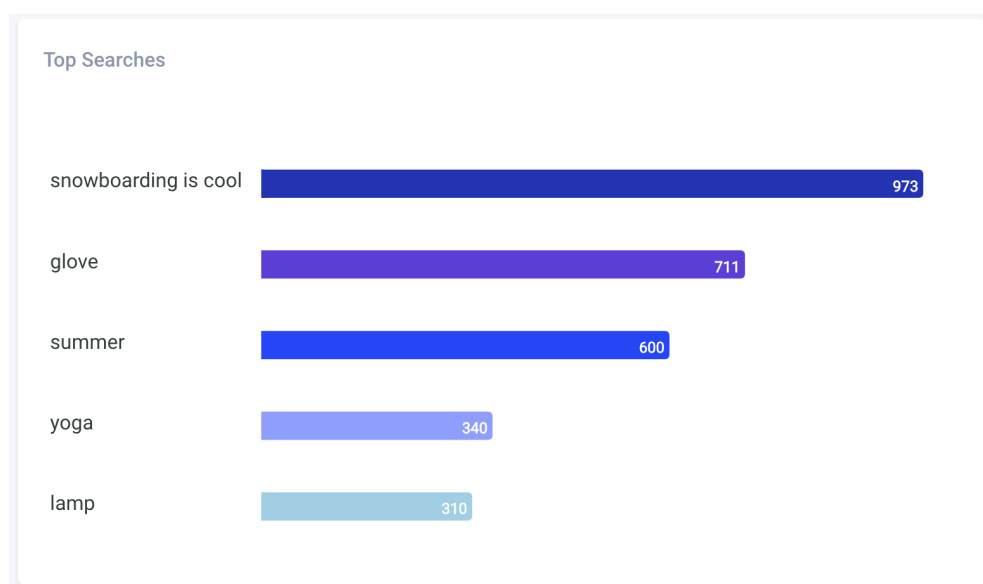
Total search click-through rate

The total search click-through rate (CTR) for a given search term. This is calculated as the number of clicks divided by the number of searches.



Top searches

Top searches are the search terms that have received the most queries from end users over the specified interval. These search terms show an overall view of what users are looking for.



Details

Click the **Top Searches** card to display details.

Top Searches

X

1H 1D 1W 1M 1Y

From Date

To Date

Search Terms	Searches	Actions	Last Created Action
Q test query	7	<div></div>	
Q *.*	6	<div></div>	
Q query3	2	<div></div>	
Q query1	2	<div></div>	
Q query2	2	<div></div>	

Field	Definition
Search Terms	The name of the query that was searched. Click this link to open a new Experience Optimizer tab that displays the query information.
Searches	The number of searches executed.
Actions	The total number of actions performed on the results. If the bar is two different colors, the left portion is the number of blocks and the right portion is the number of boosts. Click the left color to view the number of blocks or the right color to view the number of boosts.
Last Created Action	The most recent date an action was created for this search term in <div>mm/dd/yyyy</div> format and the user who initiated the action.

Most searches with low results

These search terms have received many queries from end users but relatively few results when compared to other search terms.

The chart displays the number of searches for that term on the left, and the number of results for that search on the right.



Details

Click the **Most searches with Low Results** card to display details.

Click the **Actions > Blocks** section to display the number of blocks.

Low Result Searches ✕

1H 1D 1W 1M 1Y From Date To Date

Search Terms	Searches	Hits	Actions	# Blocks: 1	Last Created Action
🔍 cypress trees	10	<div></div>	3	<div></div>	1/12/2023 (User Name)
🔍 elm trees	7	<div></div>			

Click the **Actions > Boosts** section to display the number of boosts.

Low Result Searches ✕

1H 1D 1W 1M 1Y From Date To Date

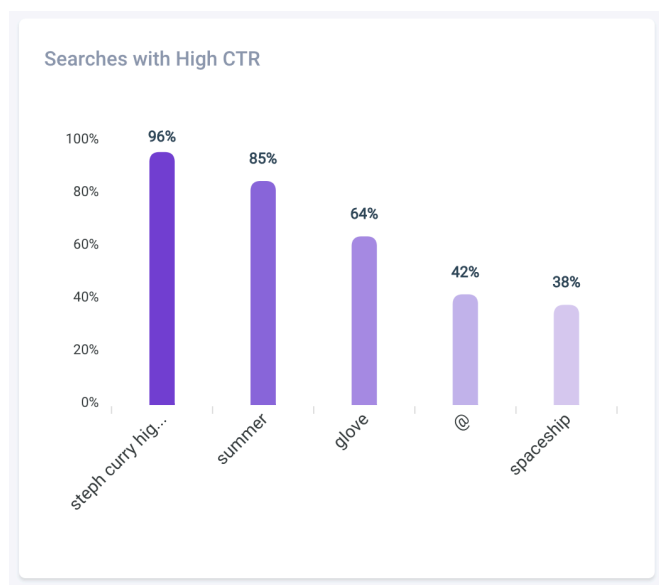
Search Terms	Searches	Hits	Actions	# Boosts: 2	Last Created Action
🔍 cypress trees	10	<div></div>	3	<div></div>	1/12/2023 (User Name)
🔍 elm trees	7	<div></div>			

Field	Definition
Search Terms	The name of the query that was searched. Click this link to open a new Experience Optimizer tab that displays the query information.
Searches	The number of searches executed.
Hits	The number of results.
Actions	The total number of actions performed on the results. If the bar is two different colors, the left portion is the number of blocks and the right portion is the number of boosts. Click the left color to view the number of blocks or the right color to view the number of boosts.
Last Created Action	The most recent date an action was created for this search term in <code>mm/dd/yyyy</code> format and the user who initiated the action.

Searches with high click-through rate

These search terms have a high click-through rate compared to other search terms. This is a sign that the results for that search term are consistently relevant to the end user's query.

A high click-through rate indicates the search results are relevant, and users are clicking on the search results frequently.

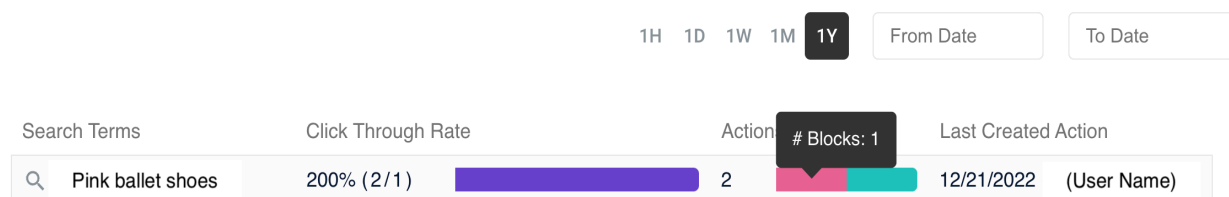


Details

Click the **Searches with High CTR** card to display details.

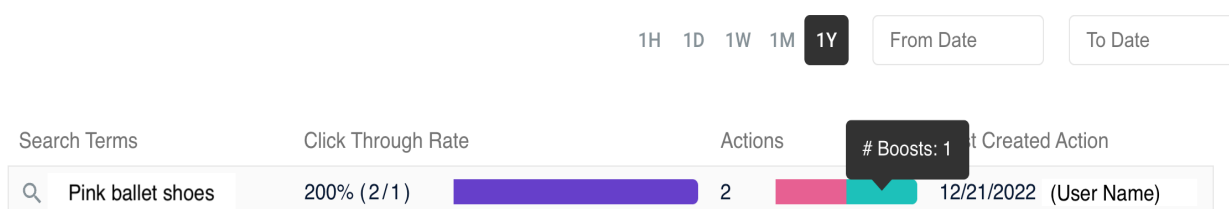
Click the **Actions > Blocks** section to display the number of blocks.

High Ctr Searches



Click the **Actions > Boosts** section to display the number of boosts.

High Ctr Searches

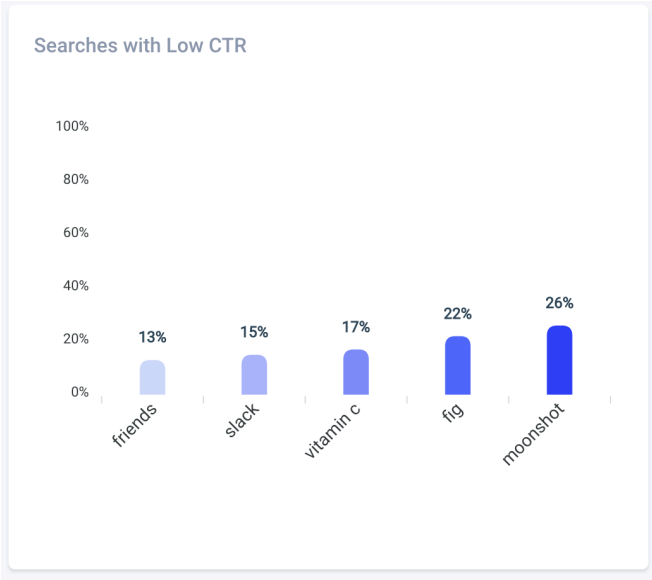


Field	Definition
Search Terms	The name of the query that was searched. Click this link to open a new Experience Optimizer tab that displays the query information.
Click Through Rate	The percentage click through rate and the (number of searches/number of clicks).
Actions	The total number of actions performed on the results. If the bar is two different colors, the left portion is the number of blocks and the right portion is the number of boosts. Click the left color to view the number of blocks or the right color to view the number of boosts.
Last Created Action	The most recent date an action was created for this search term in mm/dd/yyyy format and the user who initiated the action.

Searches with low click-through rate

These search terms have a low click-through rate compared to other search terms. This is a sign that the results for that search term are not consistently relevant to the end user's query.

A low click-through rate indicates the search results are not as relevant because users do not click on the results returned. To return more relevant results, use boost actions, block actions, and longer term signals.



Details

Click the **Searches with Low CTR** card to display details.

Low Ctr Searches

×

1H 1D 1W 1M 1Y

From Date

To Date

Search Terms	Click Through Rate	Actions	Last Created Action
test query	57% (4/7)		

Field	Definition
Search Terms	The name of the query that was searched. Click this link to open a new Experience Optimizer tab that displays the query information.
Click Through Rate	The percentage click through rate and the (number of searches/number of clicks).
Actions	The total number of actions performed on the results. If the bar is two different colors, the left portion is the number of blocks and the right portion is the number of boosts. Click the left color to view the number of blocks or the right color to view the number of boosts.
Last Created Action	The most recent date an action was created for this search term in <div>mm/dd/yyyy</div> format and the user who initiated the action.

Content with high click-through rate

These documents and files have the highest click-through rate, regardless of the search term. This content is engaging and relevant to end users across a variety of queries.

Content With High CTR

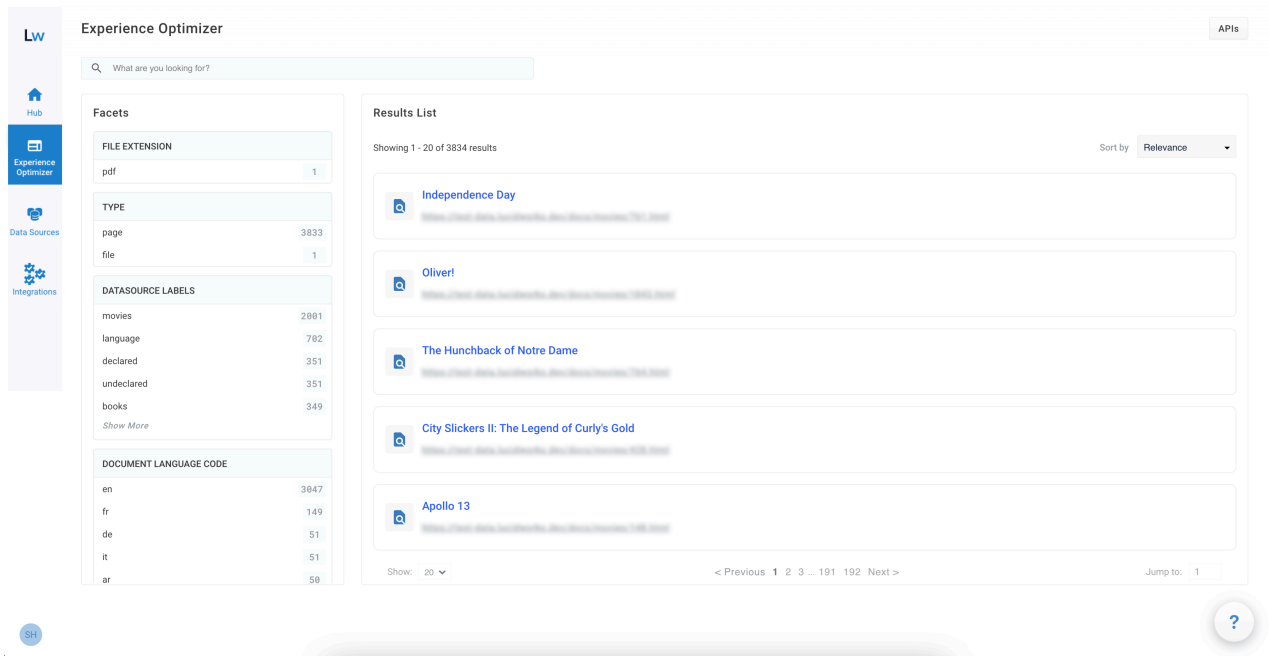
	Title 1 https://doc.lucidworkslucidworkslucidwor...	83%
	Title 2 https://doc.lucidworkslucidworkslucidwor...	78%
	Title 3 https://doc.lucidworkslucidworkslucidwor...	67%
	Title 4 https://doc.lucidworkslucidworkslucidwor...	67%
	Title 5 https://doc.lucidworkslucidworkslucidwor...	59%

Experience Optimizer

Applications UI

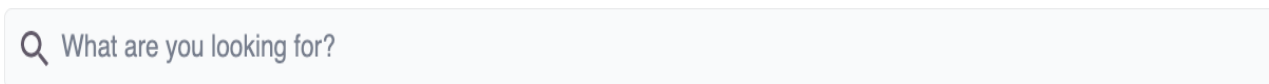
Concept

Experience Optimizer is the results configuration area for your application. Experience Optimizer lets you test, visualize, and improve search results so your end users have the best search experience.



Search bar

Enter your query in the search bar. The search bar supports queries up to 200 characters in length.



Facets

Facets display to the left of the results. Facets let you filter your search results on fields such as file type, data model type, data source labels, language code, and metadata tags.

Multiple facets can display for data sources.

- For information about data source field settings that can affect facets, see *Web data source*.
- For examples about push data source facets, see *Push data source results in Experience Optimizer*.

Facets

TYPE

Page

134

DATASOURCE LABELS

bbc

134

DOCUMENT LANGUAGE CODE

en

102

es

2

fa

2

ur

2

vi

2

Show More

PAGE METATAGS PUBLISHEDYEAR	
2016	18
2014	10
2000	8
2011	8
2013	8
<i>Show More</i>	

PAGE METATAGS PUBLISHEDDECADE	
2010s	68
1980s	8
2000s	8

Results

The results for your query display on the right side. If you haven't entered a query, all your results display here.


You can sort the results by Newest Published, Oldest Published, or Relevance.

To view *detailed information about an individual result*, click the title of that result.

Results List


Showing 1 - 20 of 3834 results

Sort byRelevance




Independence Day

https://www.imdb.com/title/tt0107290/




Oliver!

https://www.imdb.com/title/tt0060190/




The Hunchback of Notre Dame

https://www.imdb.com/title/tt0107290/



City Slickers II: The Legend of Curly's Gold

https://www.imdb.com/title/tt0107290/



Apollo 13

https://www.imdb.com/title/tt0107290/

Show:20

< Previous123...191192Next >

Jump to:1

Page details

When you select the title of an individual result, the Page Details panel displays.

Page Details



Independence Day

Excerpt

On July 2, a giant alien mothership enters orbit around Earth and deploys several dozen saucer-shaped 'destroyer' spacecraft that quickly lay waste to major cities around the planet. On July 3, the United States conducts a coordinated counterattack that fails. On July 4, a plan is devised to gain access to the interior of the alien mothership in space, in order to plant a nuclear missile. Independence Day Info imdb_id release_date language tt0116629 1996-06-25T00:00:00.000Z en

Metadata

Data Model Type Page

Date Indexed January 31, 2022 at 2:43 PM

Language Code en

Related Content

Showing 1 - 9



Fathers' Day



Roman Holiday



April Fool's Day

- **Document title.** The name of the document.
- **Document URL.** The absolute URL of the document, including hypertext protocol.
- **Excerpt.** A short section of information about the document.
- **Metadata.** Information about the content of the result.
 - **Data model type.** The data type of the document. For example, page.



Note

The only supported value for **Data model type** in push data sources is **Content**. For more information, see *Push data source results in Experience Optimizer*.

- **Date indexed.** The date and time the document was indexed. For example, December 1, 2021 at 2:34 PM. If a date cannot be determined, the default year is 1969.
- **Language code.** The abbreviation code of the document's language. For example, en for English or fr for French.
- **Key:value.** For results associated with a push data source, if **key:value** pair metadata exists, each key name and associated value is displayed. For more information, see *Push data source results in Experience Optimizer*.

- **Related Content.** Results that are semantically similar to the title and description of the entry currently selected. For example, if the movie document title is Independence Day, related content results include movies with the words that are listed in the title or description such as day, independence, or July.

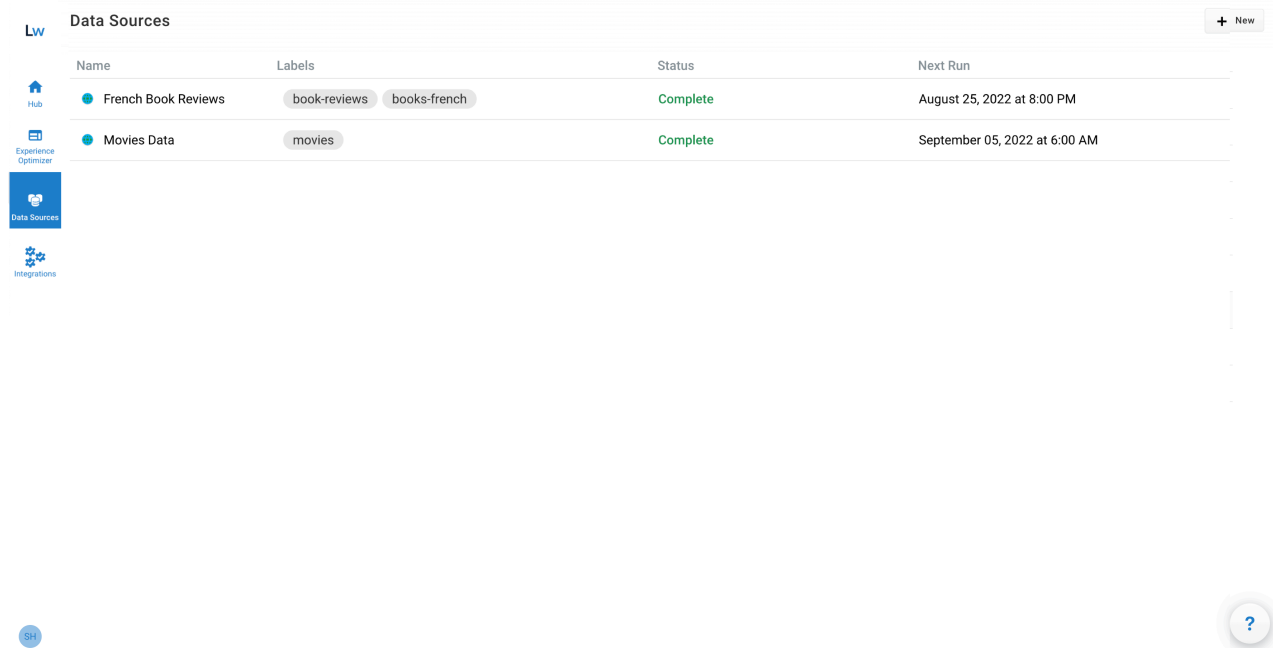
For more information about the fields, see *Query API*.

Data Sources

Applications UI

Concept


A data source is a configuration that manages the import and indexing of data into an application. In the Springboard UI, data sources contain the information you want Springboard to search. For conceptual information, see *Data sources*.



Data Sources				+ New
Name	Labels	Status	Next Run	
French Book Reviews	book-reviews books-french	Complete	August 25, 2022 at 8:00 PM	
Movies Data	movies	Complete	September 05, 2022 at 6:00 AM	

To open the screen, select the Data Sources icon from the *Applications UI sidebar* or **Manage** from the Data sources section of the *Hub screen*.

The Data Sources screen displays a list of existing data sources with the following information:

Field	Description
Name	The unique name that identifies the data source configuration in that Springboard application.
Labels	Optional short identifier. The first label is used when sorting the data sources by the Labels column.
Status	<p>The current status of the data source. To view the Last run date and time, point to the status text. The statuses are:</p> <ul style="list-style-type: none"> • Creating. The data source is being created. • Queued. The data source is in the queue for data ingestion. • Running. The data source is ingesting the data. • Complete. The data source has completed ingesting data. • Failed. The data source encountered an error during processing. • Authentication Failed. The authentication credentials for the data source are incorrect, and the data source could not be crawled. • Cleanup Skipped. If a crawl detects changes that reduce the overall size of the data source by 30% or more, the crawl errors out and the deletion process does not proceed. For more information, see <i>Existing URL crawl logic</i>. • Active. This status only applies to push data sources when the data source has been created successfully and is ready to receive data.
Next Run	<p>The date and time of the next scheduled crawl of the data source. The format is MONTH DAY, YEAR at hh:mm AM/PM in the time zone of the user's browser session. For example, November 21, 2022 at 8:30 AM.</p> <div>  Note </div> <p>The Next run field is empty for push data sources because it is not applicable.</p>

Note

If you haven't configured any data sources yet, a prompt to configure your first data source displays.

To view detailed information or edit or delete the data source, point to the entry in the list and click **View/Edit**. The Edit Data Source screen displays.

Details tab

The Details tab displays the current configuration. For more information about how to edit or delete the data source, see *Manage Springboard data sources*.

History tab

History tab for non-push data sources

For web and other non-push data sources, the History tab displays the 20 most recent data source crawl entries.



Note

Click the arrow beside a column title to change the sort order.

Field	Description
Job ID	The unique identifier for the data source crawl job. You can click the Job ID to display the <i>Job ID detailed run report</i> .
Start Time	The date and time the crawl started based on the user's browser timezone settings. The format is MONTH DAY, YEAR at hh:mm AM/PM . For example, January 21, 2022 at 6:12 PM.
Duration	If the status of the crawl is Finished , Failed , Authentication Failed , or Cleanup Skipped , the column displays the length of time from Start Time until the crawl finished or generated the error. The format is hh:mm:ss . The column displays an ellipsis ... if the status is Running .
Status	<p>The current status of the data source crawl. The statuses are:</p> <ul style="list-style-type: none">• Running. The current data source crawl is in progress.• Finished. The specific occurrence of the data source crawl finished.• Failed. The data source encountered an error during processing.• Authentication Failed. The authentication credentials for the data source are incorrect, and the data source could not be crawled.• Cleanup Skipped. If a crawl detects changes that reduce the overall size of the data source by 30% or more, the crawl errors out and the deletion process does not proceed. For more information, see <i>Existing URL crawl logic</i>.

Job ID detailed run report

The Job ID detailed run report only applies to non-push data source crawls, and provides job information about pages or files that were:


- Processed and indexed
- Excluded because the crawl encountered an access, request, or response error
- Skipped due to an issue that prevented the page or file being indexed



Tip

One important benefit of this report is that you can investigate why specific documents were not indexed, and make any necessary corrections.

Field	Description
Job ID	The unique identifier for the data source crawl job.
Back arrow	Displays the History tab for the data source.
Search	<p>Enter a term or phrase from the URL, Referrer URL, or Message columns and press Enter to display the page and file entries containing that text.</p> <p>To clear the Search field, click X. When the field is cleared, all entries for that report display.</p>
Crawl Status filter	Select a status to display only those entries. Options are: All, Error, Indexed, and Skipped. All displays every Error, Indexed, and Skipped page and file entry for that Job ID.
Crawl Status	<p>Indicates the status of the page or file entry for that Job ID. Statuses are:</p> <ul style="list-style-type: none"> • Error. Failed due to an error. For example, if the page or file cannot be discovered, a 404 Not Found error displays. • Indexed. The page or file processed and indexed without errors. • Skipped. The page or file was not processed, and is excluded from search engine results. For example, the page or file is listed as an excluded link or is not in an allowed domain.
URL	The URL the job crawls.
Referrer URL	The URL where the page or file entry was discovered.

Field	Description
Message	<p>Information about the page or file and any processing details pertinent to the entry.</p> <ul style="list-style-type: none"> Error status messages include response codes such as 404. Not found, 429 – Too many requests, and 5xx messages. Indexed status messages include information such as page or done-processing. Skipped status messages include: <ul style="list-style-type: none"> No index. The crawl determined the entry should not be indexed. No follow. The crawl determined that search engines should not follow the links on this entry. Exclude links. This entry is associated with a value entered in the Exclude links field. Missing title. The entry's document.title does not exist. Out of domain. The entry is in a domain excluded from the crawl based on the data source configuration. Size exceeded. The total content size of the entry is larger than 1 million bytes. Crawl depth exceeded. The entry depth level is higher than the value in the Limit crawl levels field. <div>  Note <p>The message may also specify a URL associated with the reason the crawl skipped the entry.</p> </div>
Page selection	Select Previous , Next , or a specific page of results.
Jump to	Select a specific result page number to display.

History tab for push data sources

For push data sources, the History tab displays the 20 most recent entries for batches that are submitted for processing.

Note

Click the arrow beside a column title to change the sort order.

Field	Description
Job ID	The unique job identifier for the specific batch associated with the push data source.
End Time	The date and time the push data source batch either failed or was successfully processed.
Status	<p>The status of the specific batch associated with the push data source. The statuses are:</p> <ul style="list-style-type: none"> • Finished. Processing for a specific batch associated with the push data source finished. • Failed. One or more errors were detected in the specific batch associated with the push data source and the batch could not be processed.

Integrations

Applications UI

Concept

To access the Integrations screen, click the Integrations icon from the Applications UI sidebar.

The screen lets you access the configuration tabs for:

- **Embeds.** Springboard embeds are drop-and-go search UI components that help you build your frontend search interface. Embeds are preconfigured and production-ready.
For more information, see *Embeds*.
- **APIs.** This tab provides links to API documentation, as well as the credentials required to obtain authentication and authorization, and perform queries in Springboard.
For more information, see *API credentials*.

Applications

Concept

Springboard applications offer tailored search experiences to specific groups of users. See *Springboard UI* for more information.

Connected Search

Connected Search is designed for site search. Index and search high-volume sites within minutes with built-in relevancy driven by machine learning.

Multiple applications

You can create multiple applications for different purposes. For example, an e-commerce company may create one application to index a product catalog, while another application only indexes an internal employee site. You can view all of your applications when you sign in to Springboard.

Status badges

Springboard uses color-coded notifications to indicate the status of an application.

An active, configured application does not display a status in the top right corner of the application. Applications that are not active display a status. For example:

- **Ready to Configure.** The application has been created and is ready to connect to a data source.
- **Service Issue.** The application has encountered an issue and is not currently functional.

Regions

Springboard supports regions for each application. Regions allow you to choose where your application data is stored and queried. You can place your Springboard applications in the geographic region closest to your end users or choose a region that helps you adhere to any regulatory requirements.

Application access

Springboard provides application security by restricting access via user authentication and authorization. You can define security on a per-application basis.

- **Authentication.** Users sign in with a username and password.
- **Authorization.** Permissions are configured to allow or restrict user access to specific functions.

Additional information

- For information about accessing Springboard, see *Sign in to Springboard*.
- For information about managing users, see:
 - *View users*
 - *Invite users*
 - *Edit or delete users*

Users

Concept

Users are members of your organization invited to access your *Springboard applications*.

Status badges

Springboard uses color-coded notifications to indicate the status of a user.

The following table describes the information:

Name	First and last name of the user.
Status	<p>Statuses include:</p> <ul style="list-style-type: none">• Active. User has clicked the Activate link in the email and can access the Springboard applications.• Awaiting Activation. User is invited, but has not clicked the Activate link in the email.• Deleted. User account has been deleted.• Deprovisioned. User account has been deactivated and cannot access the Springboard application.• Recovery. User account was once deprovisioned, but has been recovered for future use.• Password-expired. User must create a new password to reactivate the account.
Created	Date the invitation was created.
Workspace Role	Permission level for this user. Options are: Workspace Member and Workspace Owner.
Email	User's email address.

Roles

Springboard uses roles to give users varying levels of access to Springboard's applications and settings. Select a user to view their roles.

Learn more about roles.

Additional information

For information about managing users, see:

- *View users*
- *Invite users*
- *Edit or delete users*

Roles

Concept

To perform any action within Springboard, you must have access to the application or setting. This access is controlled by permissions.

Springboard consolidates sets of permissions into roles. Springboard has two types of roles: workspace roles and application roles. These roles give users varying levels of access to Springboard's applications and settings. A user can have different roles from application to application but can only have one workspace role.

Workspace roles

Workspace roles provide users with a set of permissions that let them view and manage all Springboard applications.

Workspace owners

Workspace owners have full access to every application in Springboard. They can also invite and remove users from Springboard.

Because Springboard lets you set user roles for each application, you should limit the number of users who are workspace owners.

Only workspace owners can perform the following tasks:

- Create and delete an application
- Invite and remove users from Springboard
- Add and remove roles from any user

Important

You may have multiple workspace owners in Springboard. *Contact support* if all your workspace owners no longer have access to Springboard.

Workspace members

Workspace members are Springboard users who are not workspace owners. By default, workspace members have no access to any applications.

Workspace owners must set members' roles for each application. As a result, workspace members may have different levels of access to different applications.

Workspace owners can set roles in each application for workspace members. With the proper role, workspace members may manage or view an application. Alternately, they may have no access to the application at all.

Application roles

Application roles provide users with a set of permissions that let them view and manage a specific Springboard application. Workspace owners can assign application roles to users.

Note

Workspace owners are app admins for all applications.

App admin

By default, workspace members have no access to any applications. Workspace owners can add users to applications as app admins to edit and manage certain applications without having full administrative access across Springboard.

App admins can perform the following tasks in an application:

- Edit all parts of that application
- Add, edit, and delete data sources
- Edit the settings, including the values for using the Springboard embeds

App viewer

App viewers can view an application, its data sources, and its usage dashboard. They cannot edit the application in any way, such as adding data sources or users to the application.

App viewers can access the application on the Applications page.

No access

If a user's role for an application is No Access, the user cannot view or edit the application. The application does not display on the Applications Manager screen.

Summary

The following table summarizes a user's roles and permissions within Springboard.



Note

The permissions for workspace members are set for each application.

Description	Workspace owner	App admin	App viewer
View specific applications	✓	✓	✓
View all applications	✓	✗	✗
Create new applications	✓	✗	✗
Edit specific applications	✓	✓	✗
Edit all applications	✓	✗	✗
Invite new users	✓	✗	✗
Remove or delete users	✓	✗	✗
Edit user access to all applications	✓	✗	✗
Edit user access to specific applications	✓	✓	✗
Create data sources	✓	✓	✗
View data sources	✓	✓	✓
View data source jobs	✓	✓	✓
View data source job history	✓	✓	✓
Edit relevancy controls	✓	✓	✗
View relevancy controls	✓	✓	✓
Edit embed access	✓	✓	✗
View embed settings	✓	✓	✓

Additional information

- For information about users, see *Users*.
- For information about managing user roles, see:
 - *Manage user roles from the Users page*
 - *Add users to an application from an application page*

Data sources

Concept

Data sources are complete data ingestion configurations that decide how to handle your data during ingestion by Springboard. When you run a fully configured data source, the result is a search-optimized indexed data set.

You can create and manage multiple data sources for any Springboard application.

FAQ

How do I manually re-index my data source?

Edit the data source and select **Save & Run** to initiate an immediate recrawl and data ingest.
For detailed information, see *Edit a web data source*.

Additional information

- For information about the user interface screen, see *Data sources screen*.
- For information about adding and managing data sources, see *Manage Springboard data sources*.

Web data source

Concept

Understanding the web data source configuration helps you understand how that data source crawls and indexes documents. You can then adjust as needed to get the desired results.

Web data source settings

These settings control how a Springboard app crawls and indexes a website.

- For conceptual information, see *Data sources*.
- For information about the user interface screen, see *Data sources screen*.
- For information about adding and managing data sources, see *Manage Springboard data sources*.

Data source name

In the **Data source name** field, enter a unique name that lets you easily identify the data source configuration from the list in that Springboard application.

Region

In the **Region** field, select a region from the drop-down menu. Your data is ingested in this region. Choosing a region allows you to place your Springboard applications in the geographic region closest to your end users. Regions also allow you to adhere to any regulatory requirements.

You cannot change a data source's region in an existing data source.

Start URL

For **Start URL**, enter the URL where the crawl begins. This can be a site, sitemap, or sitemap index. When crawling the same website using more than one data source configuration, it is possible to index the same page more than once. A page indexed by two separate data source configurations is assigned two different document IDs because the data source ID is used as part of the document ID.

The Start URL cannot be changed in an existing data source. You must delete the data source and add a new one with the correct URL.

Labels

Enter optional labels to identify your data source. You can create and arrange multiple labels. On the Data Sources screen, only the first label is used when sorting the data sources by the Labels column.

Include pages

Include pages has two options for pages to crawl:

- **Pages under the start URL.** Only pages under the root URL entered in the **Start URL** field are indexed and included in the crawl. Links for subdomains are ignored. For example, if the start URL is:

https://example.com/catalog :

- Valid links include:

- **https://www.example.com/catalog/books**
- **https://example.com/catalog/books**

- The crawl does not include these links:

- **https://example.com/books**
- **https://www.example.com/books**
- **https://books.example.com**

- **Pages on this site and its subdomains.** Pages on this site and its subdomains are indexed and included in the crawl. A subdomain's name ends with the domain's name. For example, A.B is a subdomain of B.

As another example, if the start URL is: **https://example.com**, a valid subdomain is **https://doc.example.com**. Links with the root URL are indexed; otherwise they are ignored.



If your start URL is a sitemap or sitemap index, select this option. Otherwise, the web crawl does not return any results.

Include file types

Select the file types to include in the crawl. The web crawl returns these document types when the extension suffix is in the URL or if the **content-type** is specified in an HTTP header. HTML content is always indexed. For detailed information, see *File extension processing*.

The web crawl can index these types of files with the specified extensions:

- Slide
 - .pptx
 - .odp
- PDF
 - .pdf
- Spreadsheet
 - .xls
 - .xlsx
 - .ods
- Word
 - .doc
 - .docx

Include external domains for selected file types

Enter external domains that contain the selected file types you want to include in the crawl. The format for the entry is **example.com**, not **https://example.com**. Do not use **https:** in the format for the external domain. Subdomains are automatically included, unless added to the list of *exclude links*.

REQUIREMENTS FOR FILES TO BE INCLUDED IN THE CRAWL

For a file to be included, it must adhere to *all* of the following guidelines:

- The file must exist in an external domain entered in the **Include external domains for selected file types** field or in a subdomain of the external domain.
- The file has *not* been excluded based on the values in the **Exclude links** field.
- The file's type must match a value selected in the **Include file types** field.

EXAMPLES

Example settings and values

The examples in the table use the following settings and values:

Setting	Value
Include file types	PDF
Include external domains for selected file types	example.com
Exclude links	https://dev.example.com/

With these settings, the following files are included because they meet these conditions:

- The file type is PDF.
- The domain is **example.com** or a subdomain.
- The subdomain is not in the list of excluded links.

URL	Included	Excluded
<code>https://example.com/customers.pdf</code>	✓	✗
<code>https://docs.example.com/guidelines.pdf</code>	✓	✗

Subdomains are automatically included. However, the subdomain **dev.example.com** is listed in the **Exclude links** field, so files located there are not included.

Although **example-dev.com** is not in the list of **Exclude links**, it is also not a domain listed in the **Include external domains for selected file types** field.

URL	Included	Excluded
<code>https://dev.example.com/code.pdf</code>	✗	✓
<code>https://example-dev.com/customers.pdf</code>	✗	✓

The file type DOCX is not a selected file type in the **Include file types** field. Even if a DOCX file is located at an allowed external domain, it is not ingested.

URL	Included	Excluded
<code>https://example.com/customers.docx</code>	✗	✓

Only files are ingested. An external page is not included, even though it is in an allowed external domain.

URL	Included	Excluded
<code>https://example.com/index.html</code>	✗	✓

Include metatags

Enter up to 10 alphanumeric metadata tag names that are ingested from the **<head>** of the HTML files and used to facet and filter search results.

If the metadata tags you enter exist and contain values, they are ingested in the crawl and the values for those metadata tags and the number of occurrences for each value display in the *Experience Optimizer Facets panel*. To populate the metadata tags facet information in Experience Optimizer, a *metatag API request* is performed during the crawl. The results information is cached for four hours. If another crawl is performed within that four-hour period, the API request is not performed again during that period because the request can be time-intensive and impact system performance.

For detailed information about metadata tags, see *Custom metadata tags*.

Include query parameters

Some websites use query components to organize and serve content. In the URL, these **key=value** pairs are found after the first **?** character following the host name and separated by the **&** character for multiple key/value pairs.

For example, if the Start URL is: `https://example.com/news`, then two unique web pages might be `https://example.com/news?year=2022&month=1` and `https://example.com/news?year=2022&month=2`.

You can enter relevant query parameters for your website in the **Include query parameters** field in the application's user interface when adding or editing a data source. These values are case-sensitive, so enter them in the **Include**

query parameters field exactly as they are used on the website.

The order of the query parameters entered does not matter. When the data source is actively crawled, the query parameters are parsed and sorted alphabetically. This ensures multiple query parameters are always compared in the same order for each URL path.

The parameters after the question mark in website URLs are identified, parsed, and then compared to the list of parameters specified in this field so different values in those parameters are treated as different pages. Only the query parameters entered in the **Include query parameters** field are used to identify unique web pages.

Important

Only the parameters entered in the **Include query parameters** field are included for the crawl, even though the Start URL may contain other parameters. Start URL website parameters not specified are ignored and are not used when creating an index.

EXAMPLE 1: NO QUERY PARAMETERS ENTERED

If no query parameters are specified in the **Include query parameters** field, then no parameters, (not even existing URL website parameters), are parsed during the crawl.

If no query parameters are entered, the following URLs are treated the same. Only the first of these links encountered during a crawl is added to the index:

- <https://example.com/news?recipes=cake>
- <https://example.com/news?recipes=pie>

EXAMPLE 2: QUERY PARAMETERS ENTERED DO NOT EXIST ON URL WEBSITE

If a query parameter that does not exist on the URL website is entered in the **Include query parameters** field, then the parameter is included in the crawl, but no results are returned. No URL website parameters are parsed during the crawl.

For example, a **recipes** query parameter is the only one entered in the **Include query parameters** field, but does not exist on the <https://example.com/news> site. The **recipes** parameter is included in the crawl, but no results are returned, and no URL website parameters are parsed during the crawl. The query parameters for these links are ignored:

- <https://example.com/news?Recipes=cake>
- <https://example.com/news?recipe=cake>
- <https://example.com/news?year=2022&month=1>

EXAMPLE 3: ONE QUERY PARAMETER ENTERED EXISTS ON THE URL WEBSITE

If only one query parameter is entered in the **Include query parameters** field, it is included in the crawl. If it exists on the URL website, relevant results are returned. No other URL website parameters are parsed during the crawl.

For example, the **month** query parameter is entered in the **Include query parameters** field and it exists on the URL website. It is included in the crawl, and relevant results are returned. But any other URL website parameters, such as **year**, are ignored. The following URLs are treated as the same page in the index:

- <https://example.com/news?month=1>
- <https://example.com/news?year=2022&month=1>

EXAMPLE 4: MORE THAN ONE QUERY PARAMETER ENTERED

All of the query parameters entered in the **Include query parameters** field are included in the crawl, even if they do not exist on the URL website. The parameters that exist on the website return relevant results. No other URL website parameters are parsed during the crawl.

If the **month** and **year** query parameters are both entered in the **Include query parameters** field, they are included in the crawl. If they exist on the URL website, they return relevant results. Any other website URL parameters, such as **city**, are ignored.

The following URLs are treated as the same page in the index:

- `https://example.com/news?month=1&year=2022`
- `https://example.com/news?month=1&year=2022&city=Union`

Include links

Add to the **Include links** field any full or partial URLs that you wish to include in the crawl. Only URLs matching these values are included.

Note

If you do not add links to your list of include links, the crawl will follow the parameters set in other fields, such as your *start URL* and *crawl levels*.

Important

Exclude links take priority over include links. For more information, see *Priority over include links*.

For example, you can choose to ingest your product articles while ignoring everything else by adding `https://example.com/products` to your list of include links.

URL	Included	Excluded
<code>https://example.com/products</code>	✓	✗
<code>https://example.com/products/electronics</code>	✓	✗
<code>https://example.com/blog/products</code>	✗	✓
<code>https://example.com/videos</code>	✗	✓
<code>https://example.com/contact</code>	✗	✓

You can also add `products` to your list of include links, but any URL containing the string is included. For best results, use the exact URL with the protocol (`http://` or `https://`) and with or without `www.` as applicable.

URL	Included	Excluded
<code>https://example.com/products</code>	✓	✗
<code>https://example.com/blog/products</code>	✓	✗
<code>https://example.com/videos</code>	✗	✓
<code>https://example.com/contact</code>	✗	✓

Exclude links

Add to the **Exclude links** field any full or partial URLs that you wish to exclude from the crawl. URLs matching the values set in this field are excluded. For best results, use the exact URL with the protocol (`http://` or `https://`) and with or without `www.`, as applicable.

Excluding links is similar to using a disallow in a `robots.txt` file and prevents ingestion of the documents. For more information, see *Robots*. Pages removed from a website still show up in query results unless excluding them is specified.

This is different from a *block*, which ingests the document but prevents it from showing up in query results.

Exclude a domain by adding it to the list of exclude links. For example, exclude the entire domain `example.com`.

URL	Included	Excluded
<code>https://example.com</code>	x	✓
<code>https://docs.example.com</code>	x	✓
<code>https://example.com/docs</code>	x	✓

Exclude unencrypted URLs by excluding `http:`. Encrypted URLs, which begin with `https:`, are crawled.

URL	Included	Excluded
<code>http://example.com</code>	x	✓
<code>https://example.com</code>	✓	x

Note that partial and full string matches exclude the URL. Excluding `dir` excludes all URLs that contain the string.

URL	Included	Excluded
<code>https://example.com/dir</code>	x	✓
<code>https://example.com/dir/sub</code>	x	✓
<code>https://example.com/sub/dir</code>	x	✓
<code>https://example.com/sub/dir.html</code>	x	✓
<code>https://example.com/directory</code>	x	✓
<code>https://example.com/directory/sub</code>	x	✓
<code>https://example.com/di/r</code>	✓	x

Exclude specific paths by excluding a target string, with or without `/`. For example, excluding `/subfolder/` excludes any URL containing the string.

URL	Included	Excluded
<code>https://example.com/subfolder/dir</code>	x	✓
<code>https://example.com/subfolder/list</code>	x	✓
<code>https://example.com/subfolder</code>	✓	x
<code>https://example.com/subfolders/list</code>	✓	x

Exclude certain file type formats by excluding the file extension. For example, if you want to ingest spreadsheet files, but you want to exclude open source formats, exclude `.ods`. Note that using `ods` instead of `.ods` excludes all paths that include `ods`.



Note

Your data source configuration must specify which file types to ingest. For more information, see [Include file types](#).

URL	Included	Excluded
https://example.com/spreadsheets/quarterly-sales.ods	x	✓
https://example.com/spreadsheets/quarterly-sales.xls	✓	x
https://example.com/pods/pod1.html	x	✓

PRIORITY OVER INCLUDE LINKS

Exclude links have priority over include links. If your include links list includes <https://example.com/dir>, and your exclude links list includes **private**, any URL that matches the exclude links term is excluded from the crawl.

URL	Included	Excluded
https://example.com/dir/docs/index.html	✓	x
https://example.com/dir/private/schedule.pdf	x	✓
https://example.com/dir/private-drive/index.html	x	✓
https://docs.example.com/dir/index.html	x	✓

Data ingest run scheduling

For **Data ingest run scheduling**, create a schedule to automatically run your data source ingestion. For example, if the value is Monthly and the data source was created on January 5, the data ingestion runs on the fifth day of each month after that.

By default, the data source schedule is set to Monthly with the date and time based on your browser time when the data source was added. If you set up the data source after the twenty-eighth day of the month, the date is set to 28.

If a scheduled data ingestion is in progress, Springboard ignores manual data ingestions for the same data source created by the **Save & Run** button. If a manual recrawl is in progress and a future data ingestion for the same data source is scheduled, the scheduled data ingestion begins after the manual recrawl is complete.

To initiate an on-demand data ingestion outside the regular schedule, see [run on-demand data ingestion](#).

The following options are available. These options cannot be combined.

- **Hourly.** Run a data ingestion every eight hours or every twelve hours, starting at the hour of your choice.
 - **Time of crawl.** The hour to run the first daily data source ingestion. The data source ingestion begins during the selected hour and runs at the selected interval.
 - **Interval.** The frequency of the crawls. The options are every eight hours or every twelve hours. For example, if you select 2:00 AM and an interval of every eight hours, then your data ingestion is scheduled to run at 2:00 AM, 12:00 PM, and 8:00 PM.
- **Daily.** Run a data ingestion once every day, starting at the hour of your choice.
 - **Time of crawl.** The hour to run the data source ingestion. The data source ingestion begins during the selected hour and runs every 24 hours afterward.
- **Weekly.** Run a data ingestion every week on the days and hour of your choice.
 - **Time of crawl.** The hour to run the data source ingestion. The data source ingestion begins during the selected hour on each day selected.

- **Days of week.** The days of the week to run a data source ingestion. Selecting different times for different days is not supported.
- **Monthly.** Run a data ingestion monthly, starting at the day and hour of your choice.
 - **Time of crawl.** The hour to run the data source ingestion. The data source ingestion begins during the selected hour and runs at that time on the selected day of each month.
 - **Day of month.** The day of the month to run a data source ingestion.

Important

Springboard uses 1-hour UTC offsets for dates and times in the UI. All dates and times display in your browser's local time zone.

If you're located in a country or territory with a 30-minute or 45-minute UTC offset, the uneven UTC offset may appear in the UI.

Limit crawl levels

Limit crawl levels has a draggable scale to set the maximum number of levels to crawl. The maximum crawl depth defines the maximum number of jumps between the start URL and any found links. Only links containing the root URL are considered when adding pages to a crawl.

The sitemap and sitemap index URLs do *not* appear in your search results. But the crawl levels function as follows:

- If the start URL is a sitemap index:
 - You may not see documents if the crawl level is set to 1 because a sitemap index typically contains sitemaps.
 - A crawl level of 2 indexes the URLs specified in the sitemaps.
 - For higher crawl levels, a crawl level of **n** indexes the links on the pages found during crawl level **n-1**.
- If the start URL is a sitemap:
 - A crawl level of 1 indexes the URLs linked in the sitemap.
 - A crawl level of 2 indexes the URLs linked on the pages found during crawl level 1.
 - For higher crawl levels, a crawl level of **n** indexes the links on the pages found during crawl level **n-1**.
- If the start URL is a regular URL:
 - A crawl level of 1 indexes the start URL and the pages linked on the start URL.
 - A crawl level of 2 indexes the URLs linked on the pages found during crawl level 1.
 - For higher crawl levels, a crawl level of **n** indexes the links on the pages found during crawl level **n-1**.

Crawl logic

New URL

When the data source is initially created and you click **Save & Run**, Springboard attempts to crawl the URL based on the values entered in the data source fields.

If the URL can be reached, Springboard ingests the URL content based on data source settings.

Existing URL

If the URL exists and Springboard has ingested the content before, subsequent crawls occur in two ways:

- On the schedule set by the value in the **Data ingest run scheduling** field.

- Started on demand when you access the *Edit a data source* screen and click **Save & Run**. You do not have to make changes to the data source to click **Save & Run** and invoke a recrawl.

Springboard detects changes to the web data source during each crawl. If an existing URL:

- Can be reached, Springboard ingests the latest version of the content.
- Cannot be reached, Springboard deletes existing content.

Important

To safeguard your data against deletion, the crawl stops and returns an error if it detects that changes to a website impact 30% or more of existing URLs. If these major changes to the data source are intentional, delete the existing data source and recreate it to ingest and reindex the documents.

Errors

For HTTP **4xx** family errors, an error message is sent to the Springboard job service and the page is not crawled. For error and resolution information, see *Client error codes 4xx*.

For HTTP **5xx** error information, see *Server error codes 5xx*.

Some errors activate retry attempts. For more information, see *Retry logic for 5xx and client-side timeout errors*.

For more information about web data source errors and how to fix them, see the *Web data source troubleshooting guide*.

Custom metadata tags

Web data source

Concept

The Springboard web data source allows you to ingest custom metadata tags from the `<head>` of your HTML files to *facet* and *filter* search results. These custom metadata tags do not affect search relevancy results.

Add or edit the data source to configure which metadata tags to ingest and store. Metadata tag **name** and **content** values are transformed into lowercase alphanumeric values with no special characters.

Springboard supports a maximum of 10 custom metadata tags per data source. Multivalued metadata tags are stored as a single string, regardless of their format.

Note

If the metadata tags you enter exist and contain values, they are ingested in the crawl and the values for those metadata tags and the number of occurrences for each value display in the *Experience Optimizer Facets panel*. If the **content** value of your metadata tag is empty, the web crawl does not ingest it. For example, `<meta name="METADATA_NAME" content="" />` is not ingested.

Metadata tag example:

```
<head> <meta name="METADATA_NAME" content="METADATA_CONTENT" />
</head>
```

The value of **name** is mapped as a field, and the value of **content** is mapped as the value of that field. For example, the metadata tag above is mapped in Springboard as follows:

```
{ "field": "page.metatags.METADATA_NAME", "values":
  ["METADATA_CONTENT"] }
```

Multiple pages can use the same **METADATA_NAME** value with different **METADATA_CONTENT** values. For example, pages could have a **difficulty** metadata tag. Each page contains one value for **difficulty**, such as **beginner** or **advanced**.

	Page 1	Page 2
Metadata tags	<pre><head> <meta name="difficulty" content="beginner" /> </head></pre>	<pre><head> <meta name="difficulty" content="advanced" /> </head></pre>
Ingested result	<pre>"fields" : { "page.metatags.difficulty ": "beginner"; }</pre>	<pre>"fields" : { "page.metatags.difficulty ": "advanced"; }</pre>

When ingested, the metadata field can be used in API requests to filter and facet results.

Facet example:

```
{ "facets": ["page.metatags.difficulty"] }
```

Filter example:



If the **content** value of your metadata tag is empty, the web crawl does not ingest it. For example, `<meta name="METADATA_NAME" content="" />` is not ingested.

Add or edit the data source to configure the metadata tags to ingest and store. These tags can be used to *facet and filter* search results.

```
{ "filters": [ {"field": "page.metatags.difficulty", "values":
  ["advanced"]} ] }
```

Robots

Web data source

Concept

What is a robots.txt file?

A **robots.txt** file tells a search engine which pages to crawl or not ignore on a website. A website can have crawling and indexing instructions for specific search engines by naming them as user agents. The instructions for crawling behavior primarily use allow and disallow rules. Additionally, there are page-level instructions for use in meta tags or HTTP headers.

Springboard's *web data source* is designed with two obedience behaviors for **robots.txt** files:

- Obey Allow, Disallow, and other rules found in a **robots.txt** file
- Obey robots meta tags and HTTP response headers for **noindex**, **nofollow**, and other rules

Note

Springboard does not use the **crawl-delay** rule because it is a non-standard instruction and Google does not use the rule in its own crawls.

File format

A **robots.txt** file is placed at the website top-level directory and is named **robots.txt**, for example **https://www.example.com/robots.txt**. The **robots.txt** file requires only two lines, **User-agent** and **Disallow**, but can have other instructions.

The **robots.txt** requires the following instructions:

- **User-agent**. Name of the crawler. Use ***** to indicate all crawlers.
- **Disallow**. URL string. Use **/** to indicate the entire site.

The following example blocks search engines from indexing the entire site:

```
User-agent: * Disallow: /
```

The following example blocks a specific directory:

```
User-agent: * Disallow: /archive/
```

What happens if the robots.txt is not found?

If a **robots.txt** file is not found and returns an HTTP **4xx** family error, Springboard assumes the site does not define the **robots.txt** file, defaults to allow all, and crawls everything.

For an HTTP **5xx** family error, the site might be on maintenance, or it could have some other problem. Springboard does not proceed with crawling until the server is in a good state.

Troubleshooting guide

Web data source

Reference

The *Springboard web data source* starts at a base URL and retrieves data from a website using HTTP.

This guide describes some common errors in using the web data source and how to fix them.

Note

Client-side request timeout and client-side connection timeout errors activate retry attempts. For more information, see *Retry logic for 5xx and client-side timeout errors*.

Incorrect language detection

The web data source crawl uses HTML properties to detect a document's language. The following table lists the HTML properties used to detect a page's language, ranked from highest priority to lowest.

- If a higher priority property's value is present, that value is used even if lower priority properties contradict the higher priority property's value.
- If a higher priority property is absent, the web crawl checks for the presence of the next attribute and its value.

Priority	Attribute	Description	Example
1	lang attribute as set in the html tag	The language the document is written in.	<code><html lang="es-ES"></code>
2	content-language header field	Describes the language of the document. The header field can contain multiple languages. This method is non-conforming in HTML5.	<code><meta http-equiv="content-language" content="en"></code>
3	Content-Language HTTP header	Describes the language of the intended audience as stated in RFC 7231 🔗 . This header may contain multiple languages.	<code>Content-Language: fr, en</code>
4	meta element's name value	Describes the language of the document. This method does not comply to any HTML standards.	<code><meta name="language" content="english"></code>

If the web crawl detects the wrong language for your web pages, check the HTML content for your pages and search for the HTML attributes in the preceding table.

The web crawl does not index any documents

If the **Start URL** is:

- An XML sitemap index and you choose to index only pages under the start URL, then the crawl may not return any pages. Edit your data source to index pages on the site and its subdomains.
- A sitemap index, set the crawl level to 2 or higher to see results.

The web crawl does not index all the expected pages

If the web crawl is missing pages or indexes some pages, but not others you expect:

- The crawl level may not be set high enough. For more information about crawl levels, see *Limit crawl levels*.
- Non-unique canonical URLs may exist. To correct this, ensure canonical URLs included in pages are unique. For example:
 - If a web page's `<head>` includes a canonical URL, such as `<link rel="canonical" href="https://example.com/index.html">`, then the web crawl uses this value as the `document.url`.
 - If two pages share the same canonical URL, then the web crawl treats them as duplicates and only indexes the first one encountered.

The web crawl indexes duplicate pages

If the web crawl indexes pages containing identical content in multiple URLs, take the following steps to deduplicate your content in the data source:

- Use the `robots.txt` file to block indexing of known duplicate pages, such as `/backup/` or `/archive/`.
- Include `<meta name="robots" content="noindex" />` at the top of the duplicated web pages.
- Use HTTP **301** permanent redirects when restructuring content causes redirects.
- If you are indexing different sections of your website with different data sources, edit each data source to exclude the other section.

New pages are not indexed

New pages are not automatically indexed. To manually reindex the content of a data source, edit the data source and click **Save & Run**. You can also reindex the data source without editing it if you access the Edit data source screen and click **Save & Run**. See *Edit a web data source* for step-by-step instructions.

If you do not manually reindex a data source, the **Data ingest run scheduling** field contains a value that specifies when the data ingestion runs. Any pages you've added recently are not indexed until the next time the data source is reindexed.

Some file types are not indexed

The web crawl indexes certain file types in addition to HTML content. See *Web data source functionality* to view all supported file types. For detailed information, see *File extension processing*.

The `meta` tag blocks individual pages

Important

If a file's `meta` tag data and the data source settings differ, the web crawl obeys the `meta` tag data first.

If a page includes a `meta` tag similar to:

- `<meta name="robots" content="noindex" />`, the page is not indexed. Remove this tag to index the page.
- `<meta name="robots" content="nofollow" />`, the web crawl cannot follow any links on that page and does not index those links. Remove this meta tag to index the links on the page. Individual links can also use the `nofollow` value such as ``. Remove `rel="nofollow"` to index the link.

Crawlers are blocked in the `robots.txt` file

Important

If the `robots.txt` file and the data source settings are different, Springboard obeys the `robots.txt` file first.

The `robots.txt` file may block crawlers.

- If your `robots.txt` file blocks all crawlers, add an exception for the Springboard crawler:

```
User-agent: Lucidworks-Web/1.1 Disallow:
```

- If pages in a subfolder are not indexed and you have not excluded that page from your data source, the `robots.txt` file may be blocking the subfolder. Edit the `robots.txt` file to unblock the subfolder.

The Exclude Links option ignores wildcards

The **Exclude Links** section does not support wildcards. You can use this section to exclude full or partial URLs.

The same page has different document IDs in different data sources

Every document has a unique ID based on the document, data source, and customer.

If the same website is indexed in two data sources, the two pages in the two data sources have different IDs. This behavior is intentional and maintains document uniqueness.

Internal link problems

If a page is not linked anywhere else on the website, that page is not crawled. Create a link to the page from elsewhere on the website, and the linked page will be crawled on the next scheduled site crawl.

A page is not indexed if it is located too many levels from the home page.

Note

Springboard's web data source supports crawl levels up to 10 levels deep. If you are optimizing your website for SEO, accessing any page from the home page should take four or fewer clicks.

The web data source can overlook links contained in outdated web technologies such as Adobe Flash or HTML frames.

Slow site crawls

Broken links can slow the web data source crawler. Perform regular site audits to fix or remove broken links.

UTF-8 characters do not parse properly

Springboard ingests content using UTF-8 character encoding to allow searches in multiple languages. On a web page with content-type character encoding set to UTF-8, the web crawl collects and indexes all properly encoded UTF-8 characters into **Page** fields. If a web page does not provide a **charset** content-type response header, then the Springboard default for character encoding is set as **charset=UTF-8**.

For anything other than alphanumeric English characters, searches are performed as an exact match. For other character encodings such as ISO-8859-1, Windows-1251, or EUC-KR, Springboard processes the characters as much as possible, but some characters might still show up as errors in the search results.

Invalid JSON error for queries with special characters

Some characters used in `search` and `searchahead` queries must be escaped for proper interpretation. Character escaping in JSON requests is handled using a backslash, `\`. If a query includes a backslash or double quotes, you must escape these or the JSON request is considered invalid.

This table lists which special characters can be escaped in a request.

Escape Sequence	Resulting Character
<code>\\</code>	<code>\</code>
<code>\"</code>	<code>"</code>
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	newline
<code>\f</code>	form feed
<code>\r</code>	carriage return

A backslash followed by any character not listed in this table results in an invalid JSON error message. Unicode character sequences are not accepted; instead they are passed as literal values. Wildcards are not supported.

Field mapping

Web data source

Reference

Data source metadata tags and other content values map to Springboard fields.

Springboard ingests field values in the order of preference specified. If the first preference is available, Springboard will ingest it and ignore the others. If the first preference is unavailable, Springboard attempts to ingest the second value, and so on.

Web page fields

UI field	API field	Source
Title	<code>document.title</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> <code><title>TITLE</title></code> <code><meta property="og:title" content="TITLE"></code> (last declared) Page URL
Description	<code>document.description</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> <code><meta name="description" content="DESCRIPTION"></code> <code><meta property="og:description" content="DESCRIPTION"></code> (last declared) <p>If no value exists for the fields above, a description is not included in search results for that page.</p>
URL	<code>document.url</code>	<p><code><link rel="canonical" href="URL"></code></p> <p>Canonical URLs must be unique for each page.</p>
Excerpt	<code>document.description</code> <code>document.body</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> <code><meta name="description" content="DESCRIPTION"></code> Contents of <code><body></code>
Data model type	<code>obj.type</code>	Content-Type HTTP header
Language code	<code>document.languageCode</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> <code><html lang="LANGUAGE_CODE"></code> <code><meta http-equiv="content-language" content="LANGUAGE_CODE"></code> Content-Language HTTP header <code><meta name="language" content="LANGUAGE_CODE"></code> Language analysis
Thumbnail image	<code>document.thumbnailImage</code>	<p><code><meta property="og:image" content="IMAGE_URL"></code></p> <p>If a value is declared, the thumbnail image displays in the Experience Optimizer's searchahead and results. If this image does not load, Springboard loads a fallback image.</p>
Timestamp	<code>document.sourceCreateTimestamp</code>	Last-Modified HTTP header

UI field	API field	Source
Timestamp	<code>document.sourceLastModifiedTimestamp</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> 1. Last-Modified HTTP header 2. Sitemap <code><lastmod></code> <p>The value is used in page objects listed on the sitemap, but it's <i>not</i> used in pages beyond the sitemap URLs.</p>

File fields

File types include spreadsheets, PDF files, slide decks, and Word documents.

UI field	API field	Source
Title	<code>document.title</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> 1. <code>Title</code> as found in file properties. 2. File URL
URL	<code>document.url</code>	Maps to the location of the file.
Excerpt	<code>document.body</code>	Maps to the text of file.
Data model type	<code>obj.type</code>	Determined using extension type. For example, <code>.doc</code> is identified as a <code>file</code> .
Language code	<code>document.languageCode</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> 1. <code>language</code> 2. <i>Language analysis</i>
Timestamp	<code>document.sourceCreateTimestamp</code>	Last-Modified HTTP header
Timestamp	<code>document.sourceLastModifiedTimestamp</code>	<p>This value is ingested in the following order of preference:</p> <ol style="list-style-type: none"> 1. Last-Modified HTTP header 2. Sitemap <code><lastmod></code>

File extension processing

If a file extension exists in the URL, results display that value in the response. If the field cannot be parsed from the URL, the HTTP header `Content-Type` MIME type is used to determine `file.extension` based on the mapping in the table in this section. If `file.extension` cannot be determined with either of these methods, the value is empty and it is not displayed in the response.

 **Note**

Older Microsoft Office MIME types are ambiguous, so the **file.extension** assigned during ingestion may differ from the original value. For example, original templates with **.dot** extensions use the **application/msword** MIME type and are ingested into Springboard with the **.doc** value.

MIME type	File extension
application/msword	.doc
application/vnd.openxmlformats-officedocument.wordprocessingml.document	.docx
application/vnd.ms-excel	.xls
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	.xlsx
application/vnd.ms-powerpoint	.ppt
application/vnd.openxmlformats-officedocument.presentationml.presentation	.pptx
application/pdf	.pdf

Push data source

Concept

The push data source must be configured when you want to use the *Push API* to ingest data that cannot be crawled by a web data source.

When you *add a push data source*, it creates a unique **DATASOURCE_ID**. The data source ID displays on the *Edit push data source screen*. This value is required to send *Push API* requests.

After the data is successfully received and ingested, it is searchable in Springboard and your website's search application.

Push data source settings

These settings configure the push data source.

Data source name

In the **Data source name** field, enter a unique name that lets you easily identify the data source configuration from the list in that Springboard application.

Region

In the **Region** field, select a region from the drop-down menu. Your data is ingested in this region. Choosing a region allows you to place your Springboard applications in the geographic region closest to your end users. Regions also allow you to adhere to any regulatory requirements.

You cannot change a data source's region in an existing data source.

Labels

Enter optional labels to identify your data source. You can create and arrange multiple labels. On the Data Sources screen, only the first label is used when sorting the data sources by the Labels column.

Results in Experience Optimizer

Information about documents ingested using the push data source displays in *Experience Optimizer*. Examples include:

- The **Data model type** facet reflects push results in the **content** entry.
- The **Labels** facet displays entries if the **Labels** field contains values for the push data source.
- If additional information about ingested documents applies, other facets reflect those values. For example, if **key:value** pair metadata exists and is faceted, each key name is listed as a facet and its associated value is displayed.
- The Page Details screen for a push result displays the **Data model type** of **Content**, which is the only supported value for a document associated with a push data source. In addition, if **key:value** pair metadata exists, each key name and associated value is displayed in the Metadata section of the screen.

Additional information

- For conceptual information, see *Data sources*.
- For information about the user interface screen, see *Data sources screen*.

Language support

Reference

Springboard comes equipped to crawl, index, and query 71 languages. Documents in any language are assigned a language code corresponding to the language detected through text analysis or as specified using a metadata tag found in the document. The language code can then be used for faceting when narrowing down query results.

If your website has the same content translated into multiple languages, readers can perform a search and then see query results matching their language of choice. Springboard returns documents containing the same sequence of characters as the query string input by your users, with the most relevant results listed first by default.


Language	Language code
Afrikaans	af
Aragonese	an
Arabic	ar
Asturian	ast
Belarusian	be
Breton	br
Catalan	ca
Bulgarian	bg
Bengali	bn
Czech	cs
Welsh	cy
Danish	da
German	de
Greek	el
English	en
Spanish	es
Estonian	et
Basque	eu
Persian	fa
Finnish	fi
French	fr
Irish	ga
Galician	gl
Gujarati	gu
Hebrew	he
Hindi	hi
Croatian	hr
Haitian	ht
Hungarian	hu

Language	Language code
Indonesian	id
Icelandic	is
Italian	it
Japanese	ja
Khmer	km
Kannada	kn
Korean	ko
Lithuanian	lt
Latvian	lv
Macedonian	mk
Malayalam	ml
Marathi	mr
Malay	ms
Maltese	mt
Nepali	ne
Dutch	nl
Norwegian	no
Occitan	oc
Punjabi	pa
Polish	pl
Portuguese	pt
Romanian	ro
Russian	ru
Slovak	sk
Slovene	sl
Somali	so
Albanian	sq
Serbian	sr
Swedish	sv

Language	Language code
Swahili	sw
Tamil	ta
Telugu	te
Thai	th
Tagalog	tl
Turkish	tr
Ukrainian	uk
Urdu	ur
Vietnamese	vi
Walloon	wa
Yiddish	yi
Simplified Chinese	zh-cn
Traditional Chinese	zh-tw

Developer documentation

Reference

The developer documentation offers tools to help you learn more about Springboard APIs. Springboard *authenticates* using [OAuth 2.0 client credentials grant type](#) , a standard that defines a secure protocol for protecting your data.

Springboard uses different URLs for the *authentication API* and *various operations APIs*:

Authentication API	<code>https://identity.lucidworks.com/</code>
Query operations API	<code>https://APPLICATION_ID.applications.lucidworks.com/</code>
Push operations API	<code>http://DATASOURCE_ID.datasources.lucidworks.com/</code>

Authentication API

Your application needs permission to access the *operations APIs*. The Springboard authentication token expires after one hour. Create a new authentication token after this time.

For more information about Springboard API authentication, see *Authentication API*.

Operations APIs

Operations APIs let you work with documents and other resources exposed by Springboard.

The *Query API* is used for various query operations, including getting results from search, searchahead, and other search-related endpoints.

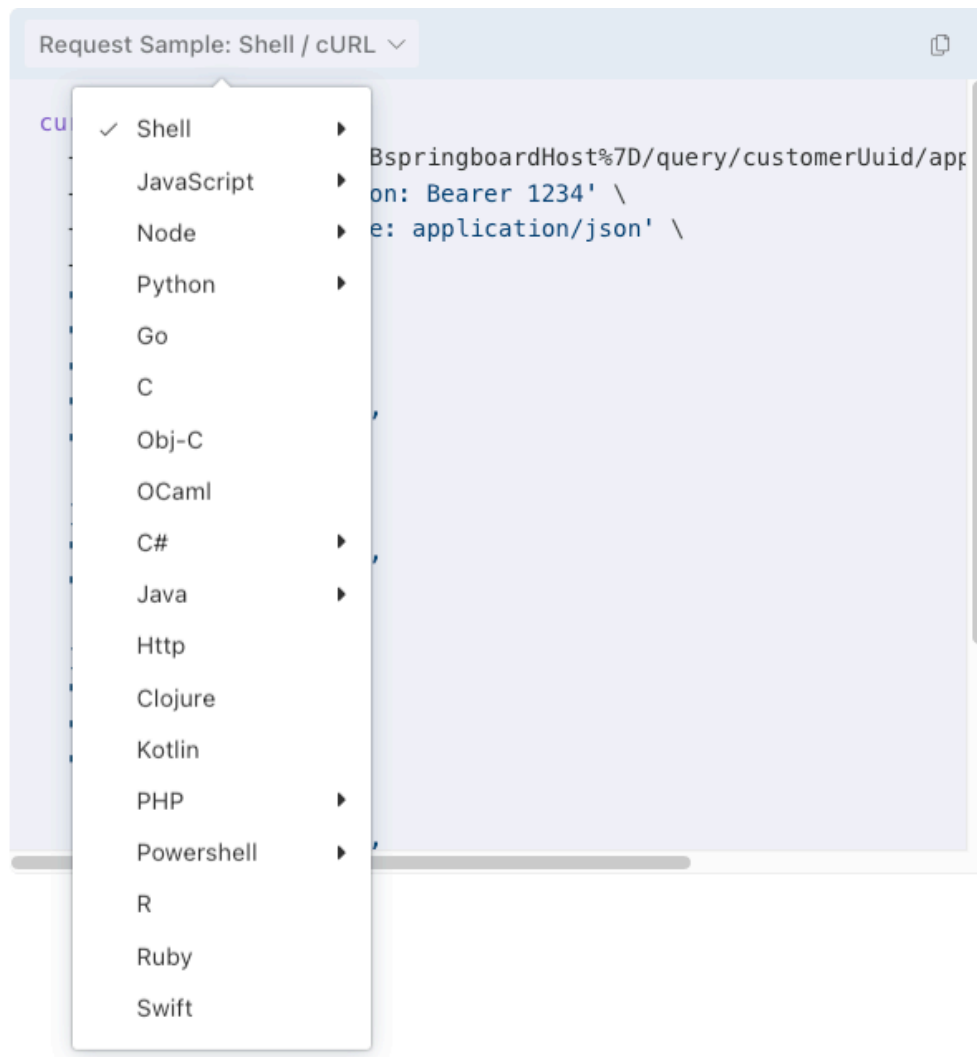
The *Push API* is used to push structured data to Springboard to make it searchable.

Developer tools

Change the request language

The developer tools offer dozens of request languages to suit your development environment. To change the language of a request:

1. Navigate to an API endpoint.
2. Click **Request Sample**.



3. From the list of languages, select a language for the API request.



Tip


To copy the request to your clipboard, click **Copy**.

Authentication API

Reference

The Authentication API authenticates a user to use the *Springboard Operations APIs*. The single endpoint can generate an access token to use in future Operations API requests.

Token guidelines and behavior

- Tokens generated by the Authentication API are [Java Web tokens \(JWT\) tokens](#) .
- Every token generated is active for 3600 seconds before it expires. After that token expires, API requests attempting to use the token fail and generate an HTTP status **401 Unauthorized** error.
- You can generate a new token at any time. To avoid authorization failures, generate a new token before the current token expires. For example, generate a new token five minutes before the current token expires.

Note

To calculate the token expiration time, Base64 decode the JWT and review the **exp** field, which contains the expiration time in seconds since the Unix epoch.

- An authentication request for a Query API token does not affect an existing Push API token.
- An authentication request for a Push API token does not affect an existing Query API token.
- You can request a token for the Query API and Push API at the same time by entering credentials and scope for both in the cURL request. A single token is generated and each API is linked to the information applicable to that API.

Note

The authentication request scope for the Query API is different than the scope for the Push API.

- Springboard uses OAuth 2.0 to authenticate your credentials.

Authentication API process

To use the Authentication API, you need the following items:

- Your Client ID, which is used for the Username value within the developer documentation tools. This is not the same as your Springboard login value.
- Your Client Secret, which is used for the Password value within the developer documentation tools. This is not the same as your Springboard login value.

To obtain the API credentials:

1. Click the Integrations icon from the *Applications UI sidebar*.
2. Click the **APIs** tab. The values are displayed on the screen.

Obtain the access token

Create a cURL request using Stoplight

1. Click the View API specification button on this page and follow the instructions listed in the Authentication API documentation.
2. For the:
 - Query API, paste the credentials in the applicable fields and use the scope value of **connectedsearch.query** to generate an access token.
 - Push API, paste the credentials in the applicable fields and use the scope value of **datasource.push** to generate an access token. For more information, see *Push API*.

Important

This authentication displays as Basic within the developer documentation tools in order to include the OAuth 2.0 authentication token that is already included. To use the request in the specification viewer, send your HTTP requests with an Authorization header that contains the word Basic followed by a space and a Base64-encoded string `CLIENT_ID:CLIENT_SECRET`.

Obtain a token using command line text

A basic token request requires the `scope` field and value.

Key	Description
<code>scope</code>	<p>The identifier that specifies the API.</p> <ul style="list-style-type: none">For Query API authentication, use the scope value of <code>connectedsearch.query</code>.For Push API authentication, use the scope value of <code>datasource.push</code>.

Important

Based on the *Token guidelines and behavior*, you must renew the current access token every hour before the 60-minute maximum expires.

To obtain or renew an access token, access a Base64 encoding tool and convert your `CLIENT_ID:CLIENT_SECRET`.

Replace `CLIENT_ID:CLIENT_SECRET` with the value you converted in the Base64 encoding tool.

```
curl --request POST \ --url
'https://identity.lucidworks.com/oauth2/ausao8uveaPmyhv0v357/v1/token?
scope=datasource.push&grant_type=client_credentials' \ --header
'Accept: application/json' \ --header 'Authorization: Basic
[CLIENT_ID:CLIENT_SECRET]' \ --header 'Cache-Control: no-cache' \ --
header 'Content-Type: application/x-www-form-urlencoded'
```

Push API

Reference

The Push API is used to push structured data to Springboard to make it searchable. Send data using the Save method to create or update documents. Use the Delete method to remove those documents.

Prerequisites

Before using this API, complete the following prerequisites:

Add the push data source

Create the push data source using the procedure in *Add a push data source*.

Obtain the Datasource ID

1. Sign in to Springboard.
2. In the *Applications Manager* screen, select the application to modify.
3. In the application's Hub screen, click the **Data Sources** icon in the left panel, or scroll to the Data Sources section of the screen and click **Manage** in the top right corner of the section.
4. In the Data Sources screen, point to the data source to edit and click **View/Edit** to the right of the entry.
5. Copy the value in the **Data source ID** field on the Details tab of the Edit Data Source screen.



For information about adding or editing a push data source, see *Manage Springboard data sources*.

Obtain an access token

Before sending a Push API request, use the *Authentication API* with the scope value of **datasource.push** to generate an access token. The access token is used to authenticate your requests.

HTTP request guidelines

The following guidelines apply to the Push API endpoints:

- The maximum HTTP request body size is 1 MB.
- If the request cannot be completed within 10 seconds, the server returns a 504 timeout error.
- Each request must contain 200 or fewer documents.

Basic POST request

The Save method uses a **POST** operation that creates or updates documents in Springboard. The request requires the **externalId**, **document.title**, and **type: content** fields and values.

Key	Description
document.title	The title of the document.
object.externalId	The unique identifier of the document. The object.externalId must be unique across all documents in the batch and can contain only lowercase ASCII characters, numbers, and hyphens.
type	Identifies the kind of information contained in the document. Specifically, it indicates the data model schema to which the document conforms. The only supported value is content .

The following example displays a minimal POST cURL request. Replace **DATASOURCE_ID** with your data source's ID. Replace **ACCESS_TOKEN** with the access token you generated using the *Authentication API*. Replace the **metadata** fields and values with any custom metadata fields and values.

```
curl --request POST
'http://DATASOURCE_ID.datasources.lucidworks.com/push' \ --header
'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '[ { "type": "content", "object": {
"externalId": "fd110486-f168-47c0-a419-1518a4840589", "metadata": {
"KEY1": "VALUE1", "KEY2": "VALUE2" } }, "document": { "title":
"title2", "description": "desc", "body": "body", "author": "author",
"createTimestamp": "2022-10-06T21:35:11Z", "lastModifiedTimestamp":
"2022-10-06T21:35:11Z", "languageCode": "en-us", "url":
"https://example.com/books/", "thumbnailImage":
"https://example.com/img.jpg" } }, { "type": "content", "object": {
"externalId": "8084969c-bd23-40f7-9acf-c68d6798bec2", "metadata": {
"FIELD1": "VALUE1", "FIELD2": "VALUE2" } }, "document": { "title":
"title22", "description": "desc", "body": "body", "author": "author",
"createTimestamp": "2022-10-06T21:35:11Z", "lastModifiedTimestamp":
"2022-10-06T21:35:11Z", "languageCode": "en-us", "url":
"https://example.com/home/", "thumbnailImage":
"https://example.com/img2.jpg" } } ]'
```

If no errors are detected in the batch, a **202 Accepted** message displays. The batch of documents posted to your Springboard application can be searched in *Experience Optimizer*. For more information, see *Push data source results in Experience Optimizer*.

Note

To view the list of responses and error messages, click **View API specification** and scroll to the response section.

Basic DELETE request

The Delete method uses a **DELETE** operation that removes a batch of documents from Springboard. The request requires the **externalId** and **type: delete** fields and values.

Key	Description
externalId	The unique identifier of the document to delete. The object.externalId must be unique across all documents in the batch and can contain only lowercase ASCII characters, numbers, and hyphens.
type	Identifies the operation being performed. Currently, the only supported value is delete .

The following example displays a minimal DELETE cURL request. Replace **DATASOURCE_ID** with your data source's ID. Replace **ACCESS_TOKEN** with the access token you generated using the *Authentication API*.

```
curl --request DELETE
'http://DATASOURCE_ID.datasources.lucidworks.com/push' \ --header
'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '[ { "type": "delete", "externalId":
"fd110486-f168-47c0-a419-1518a4840589" }, { "type": "delete",
"externalId": "8084969c-bd23-40f7-9acf-c68d6798bec2" } ] '
```

If no errors are detected in the batch, a **202 Accepted** message displays. The batch of documents deleted from your push data source can no longer be searched in *Experience Optimizer* or your website's search application.



To view the list of responses and error messages, click **View API specification** and scroll to the response section.

Processing information

Springboard validates the entire batch of documents before processing them. Save requests and delete requests handle errors differently.

Save requests

If there are no errors in any of the documents in the batch, Springboard processes the batch.

The **Save** method creates and updates documents in the same operation. If an individual document contains an **externalId** that does not currently exist in the data source, Springboard creates a new document. If an individual document contains an **externalId** that currently exists in the data source, Springboard updates the existing document.

If there are errors in any of the documents in the batch, the entire batch fails, and none of the batch is processed. The errors are logged and you must correct them before you resubmit the batch for processing.

New and modified entries may not be reflected immediately in search results.

Delete requests

If there are no errors in any of the documents in the batch, Springboard processes the batch. If an individual document contains an **externalId** that currently exists in the data source, Springboard removes the document. If no document with the specified **externalId** exists, Springboard ignores that **externalId** and the Push API does not generate an error.

If there are errors in any of the documents in the batch, the entire batch fails, and none of the batch is processed. The errors are logged and you must correct them before you resubmit the batch for processing.

Deleted entries may not be reflected immediately in search results.

Query API

Reference

The Query API is used to perform search functions on a data source.

For more information about:

- Using the Springboard user interface, see *Getting started with Springboard*.
- Adding and managing data sources, see *Manage Springboard data sources*.
- HTTP response errors, see *HTTP response errors*.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of `connectedsearch.query` to generate an access token. The access token is used to authenticate your requests.

You must also supply the `APPLICATION_ID` value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

HTTP request guidelines

The following guidelines apply to the Query API endpoints:

- The maximum HTTP request body size is 1 MiB.
- If the request cannot be completed within 10 seconds, the server returns a 504 timeout error.
- The API rate limit is 500 requests per minute per application.

Query API endpoints

Search

The results match all of the words in the query. You can also specify facets, filters, sort, and other fields to limit the results.

For example, enter the entire word "mysteries" to search the library catalog for records that include that word in the title, description, or other fields.

Information returned in the `/search` endpoint can be used in other endpoints. For example, the `id` of a result is used in the `/detail` endpoint to retrieve more information about an individual record.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/search \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "query": "Star Wars", "page": 0,
"limit": 1, "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-
8430-9e73d6006eaa", "sort": [ {"sortField": "relevancy", "sortOrder":
"asc"} ], "fieldList": ["document.title", "document.url"], "facets":
["type", "file.extension", "document.languageCode"], "highlight":
false, "filters": [ {"field": "document.languageCode", "values":
["en"]}, {"field": "datasourceLabels", "values": ["movies"]} ],
"analyticsData": true }'
```

Searchahead

The results match partial terms that are entered in a single string of characters without spaces. As with the `/search` endpoint, `/searchahead` lets you specify filters, sort, and other fields. However, the `/searchahead` endpoint does not combine with facets.

For example, results for every record that contains "nov" in the library catalog return many different genres specified as novels, or were published in November.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/searchahead \
--header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-
Type: application/json' \ --data '{ "query": "star", "page": 0,
"limit": 1, "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-
8430-9e73d6006eaa", "sort": [ {"sortField": "relevancy", "sortOrder":
"asc"} ], "fieldList": ["document.title", "document.url"],
"highlight": false, "filters": [ {"field": "document.languageCode",
"values": ["en"]}, {"field": "datasourceLabels", "values": ["movies"]}
], "analyticsData": true }'
```

Detail

Unlike the `/search` and `/searchahead` endpoints which use query terms to retrieve data, the `/detail` endpoint retrieves the details about a particular record using its `id`. You can specify more fields to display such as `document.url`, `document.author`, and `document.languageCode`.

For example, `id: "12345"` returns "The Purloined Letter" from the library catalog. Other fields you specified in the POST request are also displayed.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/detail \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "utcOffset": "-07:00",
"fieldList": ["document.title", "document.description",
"document.body", "document.sourceCreateTimestamp",
"document.sourceLastModifiedTimestamp", "document.author",
"document.url", "document.languageCode", "document.thumbnailImage",
"file.name", "file.extension"], "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa" }'
```

Metatags

Metadata tags are descriptive text placed in the **<head>** of a page's source code. The metadata tag is not displayed on the page, but can be used by search engineers and others who need that information.

For example, the metadata tag information can be used with other endpoints.

For more information, see *Web data source custom metadata tags*.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/metatags \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ }'
```

Browse

This endpoint returns all of the results for the application. You can specify facets, filters, sort, and other fields to limit the results.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/browse \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "page": 0, "limit": 2, "utcOffset":
"-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "sort":
[ {"sortField": "relevancy", "sortOrder": "asc"} ], "fieldList":
["document.title", "document.description"], "facets": ["type",
"file.extension", "document.languageCode"], "filters": [ {"field":
"document.languageCode", "values": ["en"]}, {"field":
"datasourceLabels", "values": ["movies"]} ], "analyticsData": true }'
```

Related Content

This endpoint finds and returns documents that are semantically similar. The request requires either:

- The **document.title** with the optional field of **document.description** , or
- The **id** obtained from the search result of the web page



Caution

You cannot use both the **document.title** and the **id** in the same request.

The following request using **title** includes all possible fields. This includes an example *custom metadata tag*, **page.metatags.rating**, which is requested as a value of the **fieldList** parameter.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/relatedContent \
--header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "probe": { "title": "Lost Horizon", "description": "Paradise" }, "utcOffset": "-05:00",
"fieldList": [ "document.url", "document.title",
"document.description", "document.sourceLastModifiedTimestamp",
"page.metatags.rating" ], "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "limit": 5, "source":
"https://example.com/docs/movies", "analyticsData": true }'
```

For more information and examples, see *Related Content*.

Search

Springboard Query API

Reference

The `search` endpoint of the Query API searches an indexed collection of data for relevant full-term matches to user-specified search terms. It powers the search feature in your search application.

Send your requests to the `search` endpoint using POST to create a query and receive a response.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of `connectedsearch.query` to generate an access token. The access token is used to authenticate your requests.

You must also supply the `APPLICATION_ID` value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic search request

A basic `search` request requires three fields and values: `query`, `utcOffset`, and `sessionId`.

Key	Description
<code>query</code>	The search string.
<code>utcOffset</code>	The difference between Coordinated Universal Time (UTC) and your time zone, formatted with six characters as <code>±00:00</code> .
<code>sessionId</code>	Unique value that identifies the user's session. Consider passing the session ID value from your user's browser session as the value of <code>sessionId</code> . For more information, see the MDN developer documentation for <code>sessionStorage</code> .

The following example displays a minimal `search` cURL request.

Replace `APPLICATION_ID` with your Application ID. Replace `ACCESS_TOKEN` with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/search \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "query": "Star Wars", "utcOffset":
"-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

The minimum response for the searched term or phrase includes a count of all possible matches as `hits`.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 4,
  "docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 57,
    "datasourceLabels" : [ "movies" ], "fields" : { "document.title" : "Star Wars", "document.url" : "https://example.com/docs/movies/257.html" } }, { "type" : "page",
    "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-fb148491-b39e-46d1-af33-44cd964d8ee0-34049685392290835527739651484332150529", "rank" : 8,
    "datasourceLabels" : [ "books" ], "fields" : { "document.title" : "Rebel power! / written by Lauren Nesworthy.", "document.url" : "https://example.com/docs/seattle-books/3077219.html" } }, { "type" : "page",
    "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-118109e5-7ec5-42bb-834d-e3cd41bba65f-47402309926353658638499384569395150502", "rank" : 5,
    "datasourceLabels" : [ "movies" ], "fields" : { "document.title" : "Return of the Jedi", "document.url" : "https://example.com/docs/movies/1168.html" } }, { "type" : "page",
    "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-d439fd0d-1edf-4982-b00c-51c94a5c0490-43870931094300738494010378876873965115", "rank" : 5,
    "datasourceLabels" : [ "movies" ], "fields" : { "document.title" : "The Empire Strikes Back", "document.url" : "https://example.com/docs/movies/1155.html" } } ] }
```

The second item in the preceding response is a page with a link to the first item, which is why the information returned does not initially appear to match the query.

Advanced search request

The following request includes all possible fields.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/search \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "query": "Star Wars", "page": 0,
"limit": 1, "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-
8430-9e73d6006eaa", "sort": [ {"sortField": "relevancy", "sortOrder":
"asc"} ], "fieldList": ["document.title", "document.url"], "facets":
["type", "file.extension", "document.languageCode"], "highlight":
false, "filters": [ {"field": "document.languageCode", "values":
["en"]}, {"field": "datasourceLabels", "values": ["movies"]} ],
"analyticsData": true }'
```

The following sample displays the full response.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 3,
"docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank" : 57,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Star Wars", "document.url" :
"https://example.com/docs/movies/257.html" } } ], "facets" : { "field"
: { "file.extension" : { "missing" : 3, "counts" : [ ] }, "type" : {
"missing" : 0, "counts" : [ { "name" : "page", "count" : 3 } ] },
"document.languageCode" : { "missing" : 0, "counts" : [ { "name" :
"en", "count" : 3 } ] } } } }
```

Notice the request had three hits, but only one record was returned. This happened because the number of responses in the request was listed as 1 maximum using `"page": 0, "limit": 1`.

Page display

Use the **page** and **limit** values to paginate your results. Using the default values, `"page": 0, "limit": 20` means you start with the first page and display results 1 through 20. If you change the values to `"page": 1, "limit": 20`, the request returns the second page with query matches 21 through 40.

When building the search experience for your application or website, you can use **hits** to calculate the total number of pages before zero results are shown as **Pages ≤ hits / limit**.

Note

If you do not specify **page** and **limit** in your request, the API assumes the default values and returns the first page with 20 results.

Change response sort order

The default for sorting when a response is returned is by highest to lowest ranks, meaning the match strength for that particular query.

To sort in a different way, include **sort** in your request. For example, **"sort": [{"sortField": "relevancy", "sortOrder": "asc"}]** sorts the response by lowest to highest ranks.

Highlight query terms

Use **"highlight": true** to add HTML emphasis tags (****) around words matching your search.

The following sample shows how the highlighted emphasis tags are applied.

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 2,
  "docs": [ { "type": "Page", "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank": 81,
  "datasourceLabels": [ "game" ], "fields": { "document.title": "
<em>Babbling</em> <em>Book</em>", "document.url":
"https://example.com/docs/seattle-books/KAR_009.html" } } ] }
```

Return only specified fields

Use **fieldList** to only return the fields that you specify. For example, to see a document's title in the response but not the URL or other fields, use **"fieldList": ["document.title"]**.

Filter using facets

Include facets in your request to return a count of documents containing that facet. This count appears at the end of the response after the documents.

For example, to show a count of documents containing the **type**, **file.extension**, and **document.languageCode** facets, include **"facets": ["type", "file.extension", "document.languageCode"]** in your request.

A sample response follows. The **docs** section is omitted for clarity.

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 2,
  "docs": [ ... ], "facets": { "field": { "file.extension": { "missing":
    2, "counts": [ ] }, "type": { "missing": 0, "counts": [ { "name":
      "Page", "count": 2 } ] }, "document.languageCode": { "missing": 0,
        "counts": [ { "name": "en", "count": 2 } ] } } } }
```

Filter using fields

Include **filters** in your request to limit the results to specific field values. For example, to include only documents written in English, include **"filters": [{"field": "document.languageCode", "values": ["en"]}]** in your request.

Analytics data

By default, **analyticsData** is set to **true** and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use **"analyticsData": false**.

Special characters

Character escaping in JSON requests to the **search** endpoint is handled using a backslash, ****. If a query includes a backslash or double quotes, you must escape these or the JSON request is considered invalid.

This table lists which special characters can be escaped in a request.

Escape Sequence	Resulting Character
\\	\
\"	"
\b	backspace
\t	tab
\n	newline
\f	form feed
\r	carriage return

A backslash followed by any other character returns an invalid JSON error message. Unicode character sequences are not accepted. Instead they are passed as literal values.

Searchahead

Springboard Query API

Reference

The `searchahead` endpoint of the Query API searches a collection of data for relevant partial-term matches to user-specified search terms. It powers the searchahead feature in your search application's UI, which displays real-time query matches as the user types. The query matches improve as the user continues typing.

What's the difference between `searchahead` and `search` ?

While `search` and `searchahead` perform similar tasks, they have some subtle differences.

The `search` endpoint searches for full-term matches. Use the `search` endpoint when your query is a complete search term and you want to view a list of results.

The `searchahead` endpoint searches partial-term matches for relevant documents as you type. Use `searchahead` when you want to see real-time possible matches in the search application UI.

The `searchahead` endpoint does not support facets. Use the `search` endpoint if you want to use facets in your API requests.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of `connectedsearch.query` to generate an access token. The access token is used to authenticate your requests.

You must also supply the `APPLICATION_ID` value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic `searchahead` request

A basic `searchahead` request requires three fields and values: `query`, `utcOffset`, and `sessionId`.

Key	Description
<code>query</code>	The search string.
<code>utcOffset</code>	The difference between Coordinated Universal Time (UTC) and your time zone, formatted with six characters as <code>±00:00</code> .
<code>sessionId</code>	Unique value that identifies the user's session. Consider passing the session ID value from your user's browser session as the value of <code>sessionId</code> . For more information, see the MDN developer documentation for <code>sessionStorage</code> 🔗 .

The following example displays a minimal `searchahead` cURL request.

Replace `APPLICATION_ID` with your Application ID. Replace `ACCESS_TOKEN` with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/searchahead \
--header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-
Type: application/json' \ --data '{ "query": "star", "utcOffset":
"-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", }'
```

The following sample displays a sample response.



For readability purposes, only the first three items display.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 216,
"docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank" : 5,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"The Stars Fell on Henrietta", "document.url" :
"https://example.com/docs/movies/195.html" } }, { "type" : "page",
"id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-
40bf68294303-68708807657148371333355819934570439731", "rank" : 5,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Star Trek III: The Search for Spock", "document.url" :
"https://example.com/docs/movies/1327.html" } }, { "type" : "page",
"id" : "441eb3be-7de6-470a-8141-e416a15c7db1-fb148491-b39e-46d1-af33-
44cd964d8ee0-34049685392290835527739651484332150529", "rank" : 5,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Star Trek IV: The Voyage Home", "document.url" :
"https://example.com/docs/movies/1328.html" } } ] }
```

Advanced searchahead request

The following request includes all possible fields.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/searchahead \
--header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-
Type: application/json' \ --data '{ "query": "star", "page": 0,
"limit": 1, "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-
8430-9e73d6006eaa", "sort": [ {"sortField": "relevancy", "sortOrder":
"asc"} ], "fieldList": ["document.title", "document.url"],
"highlight": false, "filters": [ {"field": "document.languageCode",
"values": ["en"]}, {"field": "datasourceLabels", "values": ["movies"]}
], "analyticsData": true }'
```

The following sample displays a response.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 202,
"docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank" : 5,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"The Stars Fell on Henrietta", "document.url" :
"https://example.com/docs/movies/195.html" } } ] }
```

Page display

Use the **page** and **limit** values to paginate your results. Using the default values, **"page": 0, "limit": 20** means you start with the first page and display results 1 through 20. If you change the values to **"page": 1, "limit": 20**, the request returns the second page with query matches 21 through 40.

When building the search experience for your application or website, you can use **hits** to calculate the total number of pages before zero results are shown as **Pages ≤ hits / limit**.

Note

If you do not specify **page** and **limit** in your request, the API assumes the default values and returns the first page with 20 results.

Change response sort order

The default for sorting when a response is returned is by highest to lowest ranks, meaning the match strength for that particular query.

To sort in a different way, include `sort` in your request. For example, `"sort": [{"sortField": "relevancy", "sortOrder": "asc"}]` sorts the response by lowest to highest ranks.

Highlight query terms

Use `"highlight": true` to add HTML emphasis tags (``) around words matching your search.

The following sample shows how the highlighted emphasis tags are applied.

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 2,
  "docs": [ { "type": "Page", "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank": 81,
  "datasourceLabels": [ "game" ], "fields": { "document.title": "
<em>Babbling</em> <em>Book</em>", "document.url":
"https://example.com/docs/seattle-books/KAR_009.html" } } ] }
```

Return only specified fields

Use `fieldList` to only return the fields that you specify. For example, to see a document's title in the response but not the URL or other fields, use `"fieldList": ["document.title"]`.

Filter using fields

Include `filters` in your request to limit the results to specific field values. For example, to include only documents written in English, include `"filters": [{"field": "document.languageCode", "values": ["en"]}]` in your request.

Analytics data

By default, `analyticsData` is set to `true` and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use `"analyticsData": false`.

Detail

Springboard Query API

Reference

The **detail** endpoint of the Query API retrieves metadata associated with a specific record.

Send your requests to the **detail** endpoint using POST to create a request and receive a response.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of **connectedsearch.query** to generate an access token. The access token is used to authenticate your requests.

You must also supply the **APPLICATION_ID** value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic detail request

A basic **detail** request requires the following three parameters in its body:

Key	Description
id	The raw numeric ID for the document. The id can be found in the response of search or searchahead requests.
utcOffset	The difference between Coordinated Universal Time (UTC) and your time zone, formatted with six characters as ±00:00 .
sessionId	Unique value that identifies the user's session. Consider passing the session ID value from your user's browser session as the value of sessionId . For more information, see the MDN developer documentation for sessionStorage .

The following sample displays a minimal **detail** cURL request.

Replace **APPLICATION_ID** with your Application ID. Replace **ACCESS_TOKEN** with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/detail \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "utcOffset": "-07:00",
"sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

The following sample displays an alternate request that also returns all fields contained in the record.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/detail \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "utcOffset": "-07:00",
"fieldList": ["document.title", "document.description",
"document.body", "document.sourceCreateTimestamp",
"document.sourceLastModifiedTimestamp", "document.author",
"document.url", "document.languageCode", "document.thumbnailImage",
"file.name", "file.extension"], "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa" }'
```

In either request, all fields contained in the specified record are returned.

If the record does not contain a particular field, the field name is not displayed in the response. The missing field does not generate an error.

The following sample displays a **detail** response.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "type" : "page",
  "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-
  40bf68294303-68708807657148371333355819934570439731",
  "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
  "Apollo 13", "document.body" : "The true story of technical troubles
  that scuttle the Apollo 13 lunar mission in 1971, risking the lives of
  astronaut Jim Lovell and his crew, with the failed journey turning
  into a thrilling saga of heroism. Drifting more than 200,000 miles
  from Earth, the astronauts work furiously with the ground crew to
  avert tragedy. Apollo 13 Info imdb_id release_date language tt0112384
  1995-06-30T00:00:00.000Z en", "document.url" :
  "https://example.com/docs/movies/148.html",
  "document.sourceCreateTimestamp" : "2022-01-31T19:31:34Z",
  "document.sourceLastModifiedTimestamp" : "2022-01-31T19:31:34Z",
  "document.languageCode" : "en" } }
```

Note

The format for the `document.sourceCreateTimestamp` and `document.sourceLastModifiedTimestamp` response values is: `yyyy-mm-ddThh:mm:ssZ`. The `T` is a separator required by the ISO 8601 combined date-time format. The `Z` is also required and signifies the zero time offset from Coordinated Universal Time (UTC). The raw API response provides two digits for the `seconds` portion of the field.

Detail request to return only specified fields

You can restrict the information returned in the request to specific fields in the record. For example, to request only the title, description, date created, and last modified date, use the following request example.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/detail \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "utcOffset": "-07:00",
"fieldList": ["document.title", "document.description",
"document.sourceCreateTimestamp",
"document.sourceLastModifiedTimestamp"], "sessionId": "864782f0-af36-
4dee-8430-9e73d6006eaa" }'
```

If the record does not contain a value in a field specified in the request, the field name is not displayed in the response. The missing field does not generate an error. In this example, the **document.description** field is missing.

The following example shows the response.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "type" : "page",
"id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-
40bf68294303-68708807657148371333355819934570439731",
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Apollo 13", "document.sourceCreateTimestamp" : "2022-01-
31T19:31:34Z", "document.sourceLastModifiedTimestamp" : "2022-01-
31T19:31:34Z" } }
```

Analytics data

By default, **analyticsData** is set to **true** and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use **"analyticsData": false**.

Metatags

Springboard Query API

Reference

The **metatags** endpoint of the Query API retrieves all the *custom metadata tags* associated with an application, where at least one document contains data in that metatag.

A **metatags** request does not retrieve any values associated with a metatag.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of **connectedsearch.query** to generate an access token. The access token is used to authenticate your requests.

You must also supply the **APPLICATION_ID** value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic metatags request

The **metatags** endpoint does not require any parameters in its body.

The following sample displays a minimal **metatags** cURL request.

Replace **APPLICATION_ID** with your Application ID. Replace **ACCESS_TOKEN** with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url  
https://APPLICATION_ID.applications.lucidworks.com/query/metatags \ --  
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:  
application/json' \ --data '{ }'
```

The following sample displays a **metatags** response from an application with two metatags, **page.metatags.publishedyear** and **page.metatags.publisheddecade**.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "fields" : [  
"page.metatags.publishedyear", "page.metatags.publisheddecade" ] }
```

Analytics data

By default, **analyticsData** is set to **true** and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use **"analyticsData": false**.

Browse

Springboard Query API

Reference

The **browse** endpoint enables developers to return all documents in the current application, sorted by relevancy. Additional sort and filter options are available.

Send your requests to the **browse** endpoint using POST to create a request and receive a response.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of **connectedsearch.query** to generate an access token. The access token is used to authenticate your requests.

You must also supply the **APPLICATION_ID** value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic browse request

A basic **browse** request requires two fields and values: **utcOffset** and **sessionId**.

Key	Description
utcOffset	The difference between Coordinated Universal Time (UTC) and your time zone, formatted with six characters as ±00:00 .
sessionId	Unique value that identifies the user's session. Consider passing the session ID value from your user's browser session as the value of sessionId . For more information, see the MDN developer documentation for sessionStorage .

The following sample displays a minimal **browse** cURL request.

Replace **APPLICATION_ID** with your Application ID. Replace **ACCESS_TOKEN** with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/browse \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "utcOffset": "-04:00", "sessionId":
"864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

A sample response follows.



Note

For readability purposes, only the first three items display.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 1000,
  "docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 1,
    "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
      "The Hunchback of Notre Dame", "document.url" :
        "https://example.com/docs/movies/764.html" } }, { "type" : "page",
      "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-fb148491-b39e-46d1-af33-44cd964d8ee0-34049685392290835527739651484332150529", "rank" : 1,
      "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
        "Apollo 13", "document.url" :
          "https://example.com/docs/movies/148.html" } }, { "type" : "page",
        "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-118109e5-7ec5-42bb-834d-e3cd41bba65f-47402309926353658638499384569395150502", "rank" : 1,
        "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
          "Mr. Smith Goes to Washington", "document.url" :
            "https://example.com/docs/movies/928.html" } } ] }
```

Advanced browse request

The following request includes all possible fields.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/browse \ --
header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-Type:
application/json' \ --data '{ "page": 0, "limit": 2, "utcOffset":
"-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "sort":
[ {"sortField": "relevancy", "sortOrder": "asc"} ], "fieldList":
["document.title", "document.description"], "facets": ["type",
"file.extension", "document.languageCode"], "filters": [ {"field":
"document.languageCode", "values": ["en"]}, {"field":
"datasourceLabels", "values": ["movies"]} ], "analyticsData": true }'
```

The following sample displays the full response.

```
{ "queryId" : "{placeholderQueryId}", "hits" : 553, "docs" : [ {  
  "type" : "page", "id" : "{placeholderDocumentId}", "rank" : 1,  
  "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :  
    "Kansas City" } }, { "type" : "page", "id" : "  
  {placeholderDocumentId2}", "rank" : 1, "datasourceLabels" : [ "movies"  
  ], "fields" : { "document.title" : "Lost Horizon" } } ], "facets" : {  
  "field" : { "file.extension" : { "missing" : 553, "counts" : [ ] },  
  "type" : { "missing" : 0, "counts" : [ { "name" : "page", "count" :  
    1996 } ] }, "document.languageCode" : { "missing" : 0, "counts" : [ {  
    "name" : "en", "count" : 553 } ] } } } }
```

Page display

Use the **page** and **limit** values to paginate your results. Using the default values, **"page": 0, "limit": 20** means you start with the first page and display results 1 through 20. If you change the values to **"page": 1, "limit": 20**, the request returns the second page with query matches 21 through 40.

When building the search experience for your application or website, you can use **hits** to calculate the total number of pages before zero results are shown as **Pages ≤ hits / limit**.

Note

If you do not specify **page** and **limit** in your request, the API assumes the default values and returns the first page with 20 results.

Change response sort order

The default for sorting when a response is returned is by highest to lowest ranks, meaning the match strength for that particular query.

To sort in a different way, include **sort** in your request. For example, **"sort": [{"sortField": "relevancy", "sortOrder": "asc"}]** sorts the response by lowest to highest ranks.

Return only specified fields

Use **fieldList** to only return the fields that you specify. For example, to see a document's title in the response but not the URL or other fields, use **"fieldList": ["document.title"]**.

Filter using facets

Include facets in your request to return a count of documents containing that facet. This count appears at the end of the response after the documents.

For example, to show a count of documents containing the **type**, **file.extension**, and **document.languageCode** facets, include **"facets": ["type", "file.extension", "document.languageCode"]** in your request.

A sample response follows. The **docs** section is omitted for clarity.

```
{ "queryId": "{placeholderQueryId}", "hits": 2, "docs": [ ... ],
  "facets": { "field": { "file.extension": { "missing": 2, "counts": []
}, "type": { "missing": 0, "counts": [ { "name": "Page", "count": 2 }
] }, "document.languageCode": { "missing": 0, "counts": [ { "name":
"en", "count": 2 } ] } } } }
```

Filter using fields

Include **filters** in your request to limit the results to specific field values. For example, to include only documents written in English, include **"filters": [{"field": "document.languageCode", "values": ["en"]}]** in your request.

Analytics data

By default, **analyticsData** is set to **true** and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use **"analyticsData": false**.

Related Content

Springboard Query API

Reference

This endpoint finds and returns documents that are semantically similar. The request requires either:

- The `document.title` with the optional field of `document.description`, or
- The `id` obtained from the search result of the web page

Caution

You cannot use both the `document.title` and the `id` in the same request.

Send a **POST** request to the `relatedContent` endpoint to create a query and receive a response.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of `connectedsearch.query` to generate an access token. The access token is used to authenticate your requests.

You must also supply the `APPLICATION_ID` value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

Basic `relatedContent` request

A basic `relatedContent` request requires one of the following strings that contain three fields and values.

- `probe.title`, `utcOffset`, and `sessionId`
- `probe.documentId`, `utcOffset`, and `sessionId`

Caution

The request can contain either the `probe.title` or the `probe.documentId`. You cannot use both `probe.title` and `probe.documentId` in the same request.

Key	Description
<code>probe.title</code>	The name of the document. Documents that are semantically similar (not case-sensitive) to the existing <code>document.title</code> are returned. If <code>probe.title</code> matches the <code>document.title</code> of a result on the webpage, it is filtered out and not returned in the results. Cannot be submitted in the request if <code>probe.documentId</code> is submitted.
<code>probe.documentId</code>	The ID of the document. Documents that are semantically similar to the existing <code>id</code> are returned. If <code>probe.documentId</code> matches the <code>id</code> of a result on the webpage, it is not returned in the results. Cannot be submitted in the request if <code>probe.title</code> is submitted.
<code>utcOffset</code>	The difference between Coordinated Universal Time (UTC) and your time zone, formatted with six characters as <code>±00:00</code> .
<code>sessionId</code>	Unique value that identifies the user's session. Consider passing the session ID value from your user's browser session as the value of <code>sessionId</code> . For more information, see the MDN developer documentation for sessionStorage .

Example request using `title`

The following example displays a minimal `relatedContent` cURL request using `title`.

Replace `APPLICATION_ID` with your Application ID. Replace `ACCESS_TOKEN` with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url  
https://APPLICATION_ID.applications.lucidworks.com/query/relatedContent  
 \ --header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-  
Type: application/json' \ --data '{ "probe": { "title": "Lost Horizon"  
}, "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-8430-  
9e73d6006eaa" }'
```

Example request using `documentId`

The following example displays a minimal `relatedContent` cURL request using `documentId`.

Replace `APPLICATION_ID` with your Application ID. Replace `ACCESS_TOKEN` with the access token you generated using the *Authentication API*.

```
curl --request POST \ --url  
https://APPLICATION_ID.applications.lucidworks.com/query/relatedContent  
 \ --header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-  
Type: application/json' \ --data '{ "probe": { "documentId":  
"441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-  
40bf68294303-68708807657148371333355819934570439731" }, "utcOffset":  
"-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Sample response

The minimum response returned is similar for requests using `title` or `documentId`, and includes a count of all possible matches as `hits`.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 9,
  "docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 1,
    "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
      "Event Horizon", "document.url" :
        "https://example.com/docs/movies/1519.html" } }, { "type" : "page",
      "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-fb148491-b39e-46d1-af33-44cd964d8ee0-34049685392290835527739651484332150529", "rank" : 2,
      "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
        "Lost in Space", "document.url" :
          "https://example.com/docs/movies/1725.html" } }, { "type" : "page",
        "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-118109e5-7ec5-42bb-834d-e3cd41bba65f-47402309926353658638499384569395150502", "rank" : 3,
        "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
          "The Lost Weekend", "document.url" :
            "https://example.com/docs/movies/1830.html" } }, { "type" : "page",
          "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-d439fd0d-1edf-4982-b00c-51c94a5c0490-43870931094300738494010378876873965115", "rank" : 4,
          "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
            "Lost Highway", "document.url" :
              "https://example.com/docs/movies/1408.html" } }, { "type" : "page",
            "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-1af001c0-cabc-4430-b3b1-c1d8f632e87a-34049685392290835527739651484332150529", "rank" : 5,
            "datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
              "Raiders of the Lost Ark", "document.url" :
                "https://example.com/docs/movies/1157.html" } } ] }
```

Advanced search request

The following request using **title** includes all possible fields. This includes an example *custom metadata tag*, **page.metatags.rating**, which is requested as a value of the **fieldList** parameter.

Example request using `title`

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/relatedContent
 \ --header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-
Type: application/json' \ --data '{ "probe": { "title": "Lost
Horizon", "description": "Paradise" }, "utcOffset": "-05:00",
"fieldList": [ "document.url", "document.title",
"document.description", "document.sourceLastModifiedTimestamp",
"page.metatags.rating" ], "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa", "limit": 5, "source":
"https://example.com/docs/movies", "analyticsData": true }'
```

Example request using `documentId`

The following request using `documentId` includes all possible fields. This includes an example *custom metadata* tag, `page.metatags.rating`, which is requested as a value of the `fieldList` parameter.

```
curl --request POST \ --url
https://APPLICATION_ID.applications.lucidworks.com/query/relatedContent
 \ --header 'Authorization: Bearer ACCESS_TOKEN' \ --header 'Content-
Type: application/json' \ --data '{ "probe": { "documentId":
"441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-
40bf68294303-68708807657148371333355819934570439731" }, "utcOffset":
"-05:00", "fieldList": [ "document.url", "document.title",
"document.description", "document.sourceLastModifiedTimestamp",
"page.metatags.rating" ], "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa", "limit": 5, "source":
"https://example.com/docs/movies", "analyticsData": true } }'
```

Sample response

The response returned is similar for requests using `title` or `documentId`. Five results were returned because the request contained `"limit": 5`.

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 9,
"docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 1,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Paradise Road", "document.url" :
"https://example.com/docs/movies/1443.html",
"document.sourceLastModifiedTimestamp" : "2022-01-31T19:37:18Z",
"page.metatags.rating": "3.4" } }, { "type" : "page", "id" :
"441eb3be-7de6-470a-8141-e416a15c7db1-fb148491-b39e-46d1-af33-44cd964d8ee0-34049685392290835527739651484332150529", "rank" : 2,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Event Horizon", "document.url" :
"https://example.com/docs/movies/1519.html",
"document.sourceLastModifiedTimestamp" : "2022-01-31T19:38:04Z",
"page.metatags.rating": "3.3" } }, { "type" : "page", "id" :
"441eb3be-7de6-470a-8141-e416a15c7db1-118109e5-7ec5-42bb-834d-e3cd41bba65f-47402309926353658638499384569395150502", "rank" : 3,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Lost in Space", "document.url" :
"https://example.com/docs/movies/1725.html",
"document.sourceLastModifiedTimestamp" : "2022-01-31T19:41:23Z",
"page.metatags.rating": "2.6" } }, { "type" : "page", "id" :
"441eb3be-7de6-470a-8141-e416a15c7db1-d439fd0d-1edf-4982-b00c-51c94a5c0490-43870931094300738494010378876873965115", "rank" : 4,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
"Ruby in Paradise", "document.url" :
"https://example.com/docs/movies/519.html",
"document.sourceLastModifiedTimestamp" : "2022-01-31T19:32:45Z",
"page.metatags.rating": "3.5" } }, { "type" : "page", "id" :
"441eb3be-7de6-470a-8141-e416a15c7db1-1af001c0-cabc-4430-b3b1-c1d8f632e87a-34049685392290835527739651484332150529", "rank" : 5,
"datasourceLabels" : [ "movies" ], "fields" : { "document.title" :
```

```
"Exit to Eden", "document.url" :  
"https://example.com/docs/movies/231.html",  
"document.sourceLastModifiedTimestamp" : "2022-01-31T19:28:22Z",  
"page.metatags.rating": "2.1" } } ] }
```

Return only specified fields

Use **fieldList** to only return the fields that you specify. For example, to see a document's title in the response but not the URL or other fields, use **"fieldList": ["document.title"]**.

Analytics data

By default, **analyticsData** is set to **true** and the request is included in analytical data. If you are testing a request and do not want it to appear in your analytical data, use **"analyticsData": false**.

HTTP response errors

Springboard API

Reference

The Springboard APIs use standard [HTTP response status codes](#). For example, success codes are in the **2xx** range, client error codes are in the **4xx** range, and so on.

Error responses include the title, message, and status code.

Important

Some HTTP status codes are specific to one or more endpoints, but not all. Review the API configuration specifications for complete details on which HTTP responses apply to your endpoint.

Client error codes **4xx**

400 - Invalid value 'FIELD_VALUE' in field 'query'. Query is too short. Must be at least 3 characters in length.

The **query** field requires a value of 3 or more characters. If there are less than 3 characters, this error is generated.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/searchahead \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": "st", "utcOffset": "-07:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Invalid value in field **query**. Query is too short. Must be at least 3 characters in length.

Solution

Enter a value of 3 or more characters in the **query** field. In this example, the invalid value **st** in the **query** field is less than 3 characters. Correct the value in the **query** field.

400 - Request body contains field 'FIELD_NAME' which cannot be repeated

The request body includes a field that cannot be repeated.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/search \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": "book", "utcOffset": "-04:00", "utcOffset": "-04:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Request body contains field `utcOffset` that cannot be repeated.

Solution

Remove the repeated field. In this example, remove the duplicate instance of `utcOffset`.

400 - Request contains more than one signal

The request body includes multiple signal objects, but the API only accepts one signal at a time.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/signals \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data ' [{ "signalType": "click", "documentId": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "utcOffset": "+04:00", "timestamp": 1655483226523 }, { "signalType": "click", "documentId": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "utcOffset": "+04:00", "timestamp": 1655483226987 } ]'
```

Response

Request contains more than one signal.

Solution

Submit each signal separately by sending individual POST requests.

400 - Request body contains field 'FIELD_NAME' with invalid value 'FIELD_VALUE'

Verify that your request body adheres to the formatting requirements found in the API specifications. If your request body uses a string for a field value that must be a number, this error may result. For example, alphabetical characters are included in the `utcOffset` field, which is a time zone formatted as `+hh:mm`.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/searchahead \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": "book", "utcOffset": "abc-07:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Request body contains field `utcOffset` with invalid value `abc-07:00`.

Solution

Correct the format of the value in the field.

400 - Request body contains field 'FIELD_NAME' with invalid value 'FIELD_VALUE' for application type 'APPLICATION_TYPE'

Verify that your request body adheres to the formatting requirements found in the API specifications for your application.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/signals \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "signalType": "visit", "documentId": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "utcOffset": "+04:00", "timestamp": 1655483226523 }'
```

Response

Request body contains the field `signalType` with invalid value `visit`.

Solution

Use a valid value in the field `signalType`, such as `click`.

400 - Request body contains field 'FIELD_NAME' with value 'FIELD_VALUE' that cannot be repeated

The request body is using the same field and value more than once. Remove the repeated value from your request body.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/searchahead \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": "book", "utcOffset": "-04:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "filters": [ {"field": "document.languageCode", "values": ["en"]}, {"field": "document.languageCode", "values": ["fr"]} ] }'
```

Response

Request body contains field `filter.field` with value `document.languageCode` that cannot be repeated.

Solution

Remove the repeated field. In this example, remove the duplicate instance of `document.languageCode`. You can specify multiple language codes by using an array:

```
{"field": "document.languageCode", "values": ["en", "fr"]}
```

400 - Request body contains invalid field 'FIELD_NAME'

The request body includes a field that is not recognized by the API. Verify that your request body does not have a typo in one of the field names and adheres to the list of fields supported by the API specifications.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/detail \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "is": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "utcOffset": "-07:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Request body contains invalid field `is` .

Solution

Correct the name of the field. In this example, change `is` to `id` .

400 - Request body contains invalid JSON

This error indicates the format for the JSON payload or content types are not correct.

API request processing priority. When an API request is sent, the JSON is validated *before* the content of the message. The invalid JSON error can indicate incorrect JSON formatting as well as invalid content format. For example, if an integer format is designated, but its value contains text, an invalid JSON error is generated.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/search \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": book, "utcOffset": "-07:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Request body contains invalid JSON.

Solution

Review the request to ensure the JSON formatting requirements are followed and the values for the content types are accurate. To check the JSON, it may help to use a JSON validator to find and resolve the issue. In this example, the double quotes (") are missing from the query value, `book` .

400 - Request body is missing required field 'FIELD_NAME'

The request body is missing at least one required field. Verify that your request body includes all of the required fields, as indicated by the API specifications.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/searchahead \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "query": "example", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Request body is missing required field `utcOffset` .

Solution

Add all required fields to the request body. In this example, the `utcOffset` field is missing from the request body.

404 - Document 'DOCUMENT_ID' was not found

The value submitted in the `id` field of the request was not found in the URL being searched.

Example:

Request

```
curl --request POST \ --url https://e2edc6de-ea00-404e-a58e-9c3810fb0c91.applications.lucidworks.com/query/detail \ --header 'Authorization: Bearer {placeholderAccessToken}' \ --header 'Content-Type: application/json' \ --data '{ "id": "1234578911234556799876", "utcOffset": "-07:00", "fieldList": [ "document.title" ], "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa" }'
```

Response

Document id was not found.

Solution

Correct the value in the `id` field.

Server error codes `5xx`

The 5xx server error codes indicate an issue with the server used in your request. These errors can occur with any request, and activate the *retry logic process*.

Error	Response
500	Internal Server Error
503	Service Unavailable
504	Gateway Timeout

Retry logic for 5xx and client-side timeout errors

The following errors indicate a temporary or transient issue, and activate the retry logic process:

- HTTP response code 500
- HTTP response code 503
- HTTP response code 504
- Client-side request timeout
- Client-side connection timeout

The retry process exponentially increments the delay with each retry. The maximum number of retries is 6.

Retry number	Delay until next retry attempt
1	1 second
2	2 seconds
3	4 seconds
4	8 seconds
5	16 seconds
6	32 seconds

Retry process details

If a retry attempt generates an error that does not activate another retry, the process ends. For example, if the first retry generates a 404 error, no more retries are attempted.

If a retry attempt generates a different error that activates the retry process, the process continues. For example, if the initial error returned is a 500 error and the first retry generates a client-side connection timeout, the second retry is activated.

If all retries have been attempted and the request is not successful, the response error from the last retry attempt must be analyzed to resolve the issue. For example, if the sixth retry generates a 503 error, begin troubleshooting to determine why the service is unavailable.

If the error resolution requires assistance from Lucidworks, click the **Support** link and add the steps that led to the error. Also include the API URL or page information and the request body. For more information, see *Contact support*.

Signals API

Reference

A signal is a user event related to a query or its search results. For example, when a user submits a query or clicks a result. When signals are recorded, your search relevancy improves over time. To share signals, use the Signals API.



This operation accepts one object at a time. Each signal must be sent by a separate POST operation.

For conceptual information, see *Signals*.

Prerequisites

Before using this endpoint, use the *Authentication API* with the scope value of `connectedsearch.query` to generate an access token. The access token is used to authenticate your requests.

You must also supply the `APPLICATION_ID` value, as the request URL requires it. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.

API credentials

Concept

The APIs tab on the *Integrations* screen provides API documentation and credentials information required for authentication and operations.

APIs

This section of the Integrations screen provides documentation links to the:

- *Developer documentation* for general information about Springboard APIs and developer tools.
- *Authentication API* to obtain the token to run queries in other Springboard APIs.
- *Query API* to perform search and other types of queries in Springboard APIs. In addition to the credentials, use the scope value of **connectedsearch.query** to generate the authentication token.
- *Push API* to push structured data to Springboard to make it searchable. In addition to the credentials, use the scope value of **datasource.push** to generate the authentication token.

Credentials

The following values are used to obtain authentication tokens and values sent in API queries:

- **Customer ID.** The unique Springboard value that identifies your organization.
- **Application ID.** The unique Springboard value that identifies a specific application. Each application has a different value.
- **Client ID.** The unique client value that is part of the information required to obtain OAuth 2.0 authorization.
- **Client Secret.** The private client value that is part of the information required to obtain OAuth 2.0 authorization. You *must* keep this value secret. Do not use it in public clients such as client-side applications.



Note

Contact [Lucidworks Support](#) if you need to regenerate your client secret.

Additional information

For information about:

- Springboard API authentication, see *Authentication API*.
- Performing Springboard Connected Search Query API requests, see *Query API*.
- Performing Push API requests, see *Push API*.

Integration tools

Concept

Springboard offers a variety of tools to help you integrate your application with your frontend and backend search interface, as an alternative *to developing with APIs*.

Embeds

Springboard embeds are drop-and-go search UI components that help you build your frontend search interface. Embeds are preconfigured and production-ready.

Embeds are designed for users who want to build their search interface fast, with basic configuration and customization options.

Search Store

The Search Store API client allows you to integrate your application with your search interface through token authentication, instead of developing your backend to use OAuth 2.0.

The Search Store is designed for users who want full control over their frontend search interface components and design.

Embeds

Concept

Springboard embeds are drop-and-go search UI components that help you build your frontend search interface. Embeds are preconfigured and production-ready.

All you need to do is install the embed and maintain it over time. You also have the option to create custom styles. Lucidworks handles the rest.

	You	Lucidworks
Install and maintain	✓	✗
Add custom styles (optional)	✓	✗
Write API calls	✗	✓
Perform API updates	✗	✓
Secure and protect	✗	✓
Collect signals	✗	✓

Note

Embed components are optimized for a desktop browsing experience.

Installation

Before you begin, *prepare your search application*. Once complete, you're ready to install an embed:

1. Click the Integrations icon from the *Applications UI sidebar*.
2. From the Embeds tab, click the toggle to make your application **Public**.
3. Add your domains to the **Allowed origins** field.
4. Click **Copy** to copy the embed script and style sheet. Paste it in the **<head>** of your HTML file.
5. Click **Copy** to copy the embed object code. Paste it in the **<body>** of your HTML file.
6. Customize, test, and launch.

For detailed installation instructions, see *Install a Springboard embed*.

FAQ

Can I use embed tokens to connect to multiple Springboard applications on the same site?

No, your site can only connect to one *Springboard application* at a time with embed tokens.

This also applies to subdomains of your site. If your site, **example.com**, uses embeds with your Springboard application, you can't connect **internal.example.com** to a different Springboard application using embed tokens.

Available embeds

Searchahead

The searchahead embed displays search results as you type. The searchahead embed starts showing results after entering three or more characters.

```
<lw-searchahead></lw-searchahead>
```

Search results

The search results embed displays the result returned by your application. Result cards include the title, description, and URL of the result. Sort options include relevance, modification date, and publication date.

```
<lw-results></lw-results>
```

Facets

The facets embed filters the results generated by the search results embed. The available facet options are pre-populated in the embed. To filter the results, click a facet.

```
<lw-facets></lw-facets>
```

Breadcrumbs

The breadcrumbs embed displays the facets currently used to filter search results. To clear a breadcrumb and remove the facet from your results, click the **X** next to the facet.

```
<lw-breadcrumbs></lw-breadcrumbs>
```

Components

Script

All that's required for embeds are the script and HTML object. You have the option to customize the embed with CSS.

This is an example embed script, located in the `<head>` of your HTML file:

```
<script type="module" id="lw-ui-lib" api-  
url="https://APPLICATION_ID.applications.lucidworks.com/" embed-  
token="EMBED_TOKEN" src="SCRIPT_SRC"></script>
```

The following values come preconfigured by Springboard:

- **APPLICATION_ID**. A unique value that identifies your application in Springboard. To obtain the value, click the Integrations icon from the *Applications UI sidebar* and then click the **APIs** tab. The value is in the **Application ID** field.
- **EMBED_TOKEN**. An access token that grants your site authenticated access to your Springboard application.

Caution

The **EMBED_TOKEN** is generated when you switch the Private toggle to Public. If you switch back to Private, the token is deleted. Switching the toggle to Public again creates a new authentication token, and your embed snippet must be replaced.

- **SCRIPT_SRC**. The script that powers the basic capability of the embed. Lucidworks updates and maintains this file.

The Springboard embed script includes one optional boolean field, **deep-linking**. Deep linking saves the search state in the URL so you can view the same results later with the same URL. Set **deep-linking = true** in the embed script to enable direct linking to the URL containing a search term. You can deactivate deep linking or customize deep linking fields within each embed's configuration options. You do not need to use the **deep-linking** field if you do not use the deep linking feature.

Important

Deep linking is not supported for embeds used within iframes.

If you're customizing the embed, you can link a style sheet below the script.

```
<script type="module" id="lw-ui-lib" api-  
url="https://APPLICATION_ID.applications.lucidworks.com/" embed-  
token="EMBED_TOKEN" src="SCRIPT_SRC"></script> <!-- Custom style sheet  
below... --> <link rel="stylesheet" href="CUSTOM_STYLESHEET_SRC">
```

HTML object

This is an example embed HTML object, located in the **<body>** of your HTML file:

```
<lw-searchahead></lw-searchahead>
```

You can combine embed HTML objects to create a complete search application:

```
<body> <div class="header"> <lw-searchahead></lw-searchahead> <lw-breadcrumbs></lw-breadcrumbs> </div> <div class="content"> <lw-facets></lw-facets> <lw-results></lw-results> </div> </body>
```

Configuration

Springboard embeds can be configured to meet the needs of your search integration. Simply include the configuration option and value in the HTML object.

For example, the search results embed can be configured to only return results with the tag **news**. To do this, add the **filter** configuration option to target the tag:

```
<lw-results filter="tag:news" ></lw-results>
```

Visit your embed configuration page for full configuration options.

Breadcrumbs

Embed configuration options

Reference

The breadcrumbs embed displays the facets currently used to filter search results. To clear a breadcrumb and remove the facet from your results, click the **X** next to the facet.

```
<div> <lw-breadcrumbs></lw-breadcrumbs> </div>
```



To display the breadcrumbs embed, you must also use the *facets embed*.

Configuration

Breadcrumbs embed

Reference

The breadcrumbs embed offers configuration options to customize what displays in your embed.

Configuration options

Parameter	Value s	Description
<code>facetLabels</code>	object	Controls the category name of the facets displayed. If a value is not designated, the name from the API is used.
<code>analyticsData</code>	boolean	<p>If true, the request is included in analytical data. Set <code>analyticsData</code> to false to exclude the query from the application metrics and other analytical data. Set to false for all non-production requests, including tests.</p> <p>This field does not affect the query or results.</p> <ul style="list-style-type: none">Default value: <code>true</code>

Important

If you are using the breadcrumbs embed with the *facets embed*, the `facetLabels` configuration values must be identical for the breadcrumbs and facets embeds.

Configuration example

The information passed in an HTML page must use valid JSON syntax that can be parsed, converted, and applied.

```
<lw-breadcrumbs config='{ "facetLabels": { "file.extension": "File  
Type", "type": "Document Type", "datasourceLabels": "Labels"},  
"analyticsData": false }'></lw-breadcrumbs>
```

Note

You can also send valid JavaScript object syntax. For more information about JSON formatting, see [W3Schools JSON Syntax](#).




CSS styling

Breadcrumbs embed

Reference

The breadcrumbs embed includes CSS custom properties so you can match the searchahead embed's appearance to your website's theme and branding.

Consult the following resources for general CSS guidance and tutorials:

- [W3Schools CSS tutorial](#) 
- [MDN CSS reference](#) 
- [MDN CSS custom properties](#) 

The following code sample displays the breadcrumbs embed's CSS custom variables and their default values. Click [Edit on CodePen](#) to open a new screen and save your work.

HTML

CSS

Result


EDIT ON

LIVE

```
lw-facets {
  float: left;
}

lw-breadcrumbs {
  /* Breadcrumb border CSS
   Breadcrumb border variables
   control the appearance of the
   box surrounding the
   breadcrumb.*/
  --breadcrumb-border-radius: 0;
  --breadcrumb-border-color:
#ebeced;
  --breadcrumb-border-style:
solid;
  --breadcrumb-border-width:
1px;

  /* Close button CSS
   Close icon variables control
   the appearance of the icon used
   to clear a facet.*/
  --close-icon-width: 7px;
  --close-icon-margin: 6.5px;
  --close-icon-bg-color:
#889198;
  --close-icon-border-radius: 0;
  --close-icon-bg-image:
url("data:image/svg+xml;charset=
utf-8,%3Csvg width='14'
height='14'
xmlns='http://www.w3.org/2000/sv
g'%3E%3Cpath d='M14 1.41L12.59 0
7 5.59 1.41 0 0 1.41 5.59 7 0
```



Resources1x 0.5x 0.25xRerun

Facets

Embed configuration options

Reference

The facets embed filters the results generated by the search results embed. The available facet options are pre-populated in the embed. To filter the results, click a facet.

```
<div> <lw-facets></lw-facets> </div>
```

Result



Tip

Try it!

To test the embed, complete the following:

- Click a facet.
- Combine facets from separate facet groups.
- Click a facet again to remove it.

Configuration

Facets embed

Reference

The facets embed offers configuration options to customize what displays in your embed.

Configuration options

Parameter	Values	Description
facets	array	Controls the facets displayed for the user. Only the facets designated are requested in the API.
facetLabels	object	Controls the category name of the facets displayed. If a value is not designated, the name from the API is used.
limit	integer	<p>Controls the number of facet values displayed at a time.</p> <p>For example, if limit: 20, then 20 facet values are displayed on the window at a time. If more than 20 facet values exist, clicking Show More displays 20 more facet values at time up to the maximum of 100 allowed facet values.</p> <p>In other words, if limit: 20 and 35 values exist, the first 20 are displayed and clicking Show More displays the remaining 15.</p> <p>The limit field is a global setting for all facets. Use the facetLimits field to override limit for an individual facet field.</p> <ul style="list-style-type: none"> • Default value: 5 • Maximum value: 100
facetLimits	integer	<p>Controls the maximum number of values displayed for a specific facet field at a time. This value overrides the global limit value for that specific facet field.</p> <p>For example, if limit: 20, but facetLimits: { datasourceLabels: 10 }, then datasourceLabels displays 10 at a time (instead of 20 at a time). In this example, if more than 10 values exist for datasourceLabels, clicking Show More displays 10 more facet values at time up to the maximum of 100 facet values.</p> <p>In other words, if facetLimits: { datasourceLabels: 10 } and 22 datasourceLabels values exist, the first 10 are displayed and clicking Show More displays the next 10, then clicking Show More again displays the remaining 2.</p> <ul style="list-style-type: none"> • Default value: 5 • Minimum value: 1 • Maximum value: 100
deepLinking	boolean or object	<p>Customizes the deep linking options in the embed. You can specify the URL query parameters for the embed to use. For example, if you use f to indicate the filters parameter elsewhere on your website, Springboard can continue using f as your filters parameter instead of filters. Springboard can customize the filters field name.</p> <ul style="list-style-type: none"> • If "deepLinking": false in the embed configuration, then deep linking is not enabled for that embed, and any parameters set in that embed are not included in the URL. • If deep-linking = false or does not exist in the <i>embed configuration script</i>, adding values to the deepLinking field in the embed configuration has no effect.

Parameter	Values	Description
group	string	Specifies the group that your embeds are in. A group of embeds uses the same deep linking and URL query parameter values. This parameter is outside of the config object. For example, using the facets embed in group A does not affect the embeds in group B, even if group B includes another facets embed.
analyticsData	boolean	<p>If true, the request is included in analytical data. Set analyticsData to false to exclude the query from the application metrics and other analytical data. Set to false for all non-production requests, including tests.</p> <p>This field does not affect the query or results.</p> <ul style="list-style-type: none"> Default value: true

Important

If you are using the facets embed with the *breadcrumbs embed*, the **facetLabels** configuration values must be identical for the breadcrumbs and facets embeds.

Configuration example

The information passed in an HTML page must use valid JSON syntax that can be parsed, converted, and applied.

```
<lw-facets config='{ "limit": 50,
  "facetLimits": { "datasourceLabels": 10 }, "facets":
[ "file.extension", "type", "datasourceLabels", "facetLabels":
{ "file.extension": "File Type", "type": "Document Type",
"datasourceLabels": "Labels"}, "analyticsData": false }'></lw-facets>
```

Note

You can also send valid JavaScript object syntax. For more information about JSON formatting, see [W3Schools JSON Syntax](#).




CSS styling

Facets embed

Reference

The facets embed includes CSS custom properties so you can match the facets embed's appearance to your website's theme and branding.

Consult the following resources for general CSS guidance and tutorials:

- [W3Schools CSS tutorial](#) 
- [MDN CSS reference](#) 
- [MDN CSS custom properties](#) 

The following code sample displays the facets embed's CSS custom variables and their default values. Click **Edit on CodePen** to open a new screen and save your work.

HTML

CSS

Result


EDIT ON

LIVE

```
lw-facets {
  /* This makes the embeds look
  better. */
  float: left;

  /* Facets containers CSS
  Facets variables control the
  appearance of the space
  containing all the facets.*/
  --facets-border-radius: 4px;
  --facets-border-width: 1px;
  --facets-border-style:
solid;
  --facets-border-color:
#EBECED;
  --facets-title-display:
block;
  --facets-padding: 0 16px;
  --facets-width: inherit;
  --facets-margin: unset;
  --facets-flex: 0 0 52px;

  /* Facets list CSS
  Facets list variables control
  the appearance of the list of
  facets that a user can use to
  filter results.*/
  --facets-list-padding: 16px;
  --facets-list-border-radius:
4px;
  --facets-list-overflow-x:
hidden;
  --facets-list-height: 100%;
  --facets-list-display:
```



Resources1x0.5x0.25xRerun

Related content

Embed configuration options

Reference

The related content embed displays documents that are semantically related to a query.

The related content embed also collects click signals when an end user clicks a document or URL. The embeds send these click signals to Springboard. No additional setup is required to collect signals.

```
<div> <lw-related-content></lw-related-content> </div>
```

Result



Tip

Try it!

To test the embed, enter a term or phrase in the Search field.

Configuration

Related content embed

Reference

The related content embed offers configuration options to customize what displays in your embed.

Configuration options

Parameter	Value s	Description
<code>limit</code>	num ber	The number of results to display in the related content embed. <ul style="list-style-type: none">Default value: <code>5</code>
<code>probe</code>	obje ct	Controls the parameters to use when searching for related content. Use <code>{ title: 'TITLE_SEARCH_TERM', description: 'DESCRIPTION_SEARCH_TERM' }</code> to display content related to the specified title and description. The <code>title</code> value is required. The <code>description</code> value is optional.
<code>fieldLi st</code>	obje ct	Specifies any fields to use when displaying fields in the <code>displayFields</code> object. Any fields not used by default are required.
<code>display Fields</code>	obje ct	Controls the fields that display in the related content embed. <ul style="list-style-type: none">Default value: <code>{"title": "document.title", "url": "document.url"}</code>

Display fields customization

You can use the contents of a different metadata tag to override the fields that display in the related content embed.

For every display field not included by default in the `fieldList`, you must specify that field and its value in the `displayFields` object. If you use a metadata tag for one of the display fields, that metadata tag must be indexed in your data source. If you specify a metadata tag in the configuration options and the metadata tag is not available in the document, then Springboard uses the default value for that parameter.

If you do not specify a metadata tag in the related content configuration to display, the default value is used. If no default value exists, Springboard uses a fallback value.

Parameter	Default value	Description
<code>title</code>	<code>document.t itle</code>	The title to display in the related content embed.
<code>descrip tion</code>	<code>document.d escriptio n</code>	The description to display in the related content embed.
<code>url</code>	<code>document.u rl</code>	The URL to display in the related content embed.
<code>image</code>	none	The thumbnail image to display in the related content embed. If no <code>image</code> value is provided, Springboard uses the <code>document.thumbnailImage</code> fallback value.

Configuration example

The information passed in an HTML page must use valid JSON syntax that can be parsed, converted, and applied.

```
<lw-related-content config='{ "limit": 2, "probe": { "title":  
"TITLE_SEARCH_TERM"}, "fieldList": ["document.title", "document.url",  
"CUSTOM_METADATA_TAG"], "displayFields": {  
  "title": "document.url",  
  "description": "document.title",  
  "url": "CUSTOM_METADATA_TAG" } }'></lw-related-content>
```

Note

You can also send valid JavaScript object syntax. For more information about JSON formatting, see [W3Schools JSON Syntax](#) .




CSS styling

Related content embed

Reference

The related content embed includes CSS custom properties so you can match the embed's appearance to your website's theme and branding.

Consult the following resources for general CSS guidance and tutorials:

- [W3Schools CSS tutorial](#) 
- [MDN CSS reference](#) 
- [MDN CSS custom properties](#) 

The following code sample displays the related content embed's CSS custom variables and their default values. Click **Edit on CodePen** to open a new screen and save your work.

HTML

CSS

Result


EDIT ON

LIVE

```
lw-related-content {
  /* Results box CSS
  The results variables control
  the appearance of the results
  box.*/
  --results-border: solid 1px
#ebeced;
  --results-border-radius: 4px;
  --results-padding: 0 16px;

  /* Results list CSS
  The results variables control
  the appearance of the results
  list.*/
  --results-list-overflow:
hidden;
  --results-list-padding-top:
0px;
  --results-list-border-radius:
4px;
  --results-list-bg-color:
#ffffff;
  --results-list-no-results-
height: 30rem;

  /* Results pagination CSS
  Results pagination variables
  control the appearance of the
  number of results and the
  pagination selection options.
  Results pagination is not
  currently supported in the
  related content embed.*/
  --results-pagination-display:
```



Resources1x 0.5x 0.25xRerun

Searchahead

Embed configuration options

Reference

The searchahead embed displays search results as you type. The searchahead embed starts showing results after entering three or more characters.

```
<div> <lw-searchahead></lw-searchahead> </div>
```

Result



Tip

Try it!

To test the embed, enter a term or phrase in the Search field.

Event logic

Click a result from the searchahead list, or press the **Enter** (or **Return**) key to submit the query. In either case, the searchahead embed fires a **query** event:

```
document.querySelector('lw-searchahead').addEventListener('query',  
(event) => { const detail = event.detail; console.log(detail); });
```



Note

Although the preceding example is representative of the query logic, the actual event differs between JavaScript frameworks.

The JSON payload sent by the event depends on the submission method used. For example, clicking a result from the searchahead sends a **click** trigger, as well as the document ID and the title of the result as the **query** value:

```
{ "query": "Manufacturing to Industry Standards", "trigger": "click",  
  "docId": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-  
94aa-40bf68294303-68708807657148371333355819934570439731" }
```

Pressing the **Enter** key sends an **enter** trigger and the query value typed in the searchahead embed:

```
{ "query": "manufacturing", "trigger": "enter" }
```

Configuration

Searchahead embed

Reference

The searchahead embed offers configuration options to customize what displays in your embed.

Configuration options

Parameter	Type	Description
<code>placeholder</code>	string	Controls the placeholder text for the object. This parameter is outside of the <code>config</code> string. The <i>configuration example</i> displays the appropriate syntax. <ul style="list-style-type: none"> Default value: <code>Search</code>
<code>type-debounce</code>	integer	Designates the number of milliseconds to wait before submitting the query. This parameter is outside of the <code>config</code> string. The <i>configuration example</i> displays the appropriate syntax.
<code>limit</code>	number	The maximum number of results returned per page. This does not affect the total number of documents in the result set, or "hits." <ul style="list-style-type: none"> Default value: <code>20</code>
<code>highlight</code>	boolean	Enables or disables highlighting of results. <ul style="list-style-type: none"> Default value: <code>true</code>
<code>filters</code>	array	Specifies which facets to select. The format is an array of objects where each object contains a <code>field</code> and a <code>value</code> .
<code>sort</code>	array	Specifies sort order. <ul style="list-style-type: none"> Default value: <code>relevancy asc</code>
<code>fieldList</code>	array	Specifies any fields to use when displaying fields in the <code>displayFields</code> object. Any fields not used by default are required. <ul style="list-style-type: none"> Default value: <code>["document.title", "document.thumbnailImage"]</code>
<code>displayFields</code>	object	Specifies the metadata tags to use when displaying the title and thumbnail image. If you do not specify a metadata tag in the searchahead configuration to display, Springboard uses the default value. See <i>Display fields customization</i> for more information.
<code>deepLinking</code>	boolean or object	Customizes the deep linking options in the embed. You can specify the URL query parameters for the embed to use or turn deep linking off for one embed without affecting other embeds. For example, if you use <code>s</code> to indicate the query parameter elsewhere on your website, Springboard can continue using <code>s</code> as your query parameter instead of <code>query</code> . Springboard can customize the <code>query</code> and <code>searchaheadFilters</code> field names. <ul style="list-style-type: none"> If <code>"deepLinking": false</code> in the embed configuration, then deep linking is not enabled for that embed, and any parameters set in that embed are not included in the URL. If <code>deep-linking = false</code> or does not exist in the <i>embed configuration script</i>, adding values to the <code>deepLinking</code> field in the embed configuration has no effect.

Parameter	Type	Description
group	string	Specifies the group that your embeds are in. A group of embeds uses the same deep linking and URL query parameter values. This parameter is outside of the config value. For example, using the searchahead embed in group A does not affect the embeds in group B, even if group B includes another searchahead embed.
analyticsData	boolean	<p>If true, the request is included in analytical data. Set analyticsData to false to exclude the query from the application metrics and other analytical data. Set to false for all non-production requests, including tests.</p> <p>This field does not affect the query or results.</p> <ul style="list-style-type: none"> Default value: true

Display fields customization

You can use the contents of a different metadata tag to override the fields that display in the searchahead embed.

For every display field not included by default in the **fieldList**, you must specify that field and its value in the **displayFields** object. If you use a metadata tag for one of the display fields, that metadata field must be indexed in your *data source*. If you specify a metadata tag in the configuration options and the metadata tag is not available in the document, then Springboard uses the default value for that parameter.

If you do not specify a metadata tag in the searchahead configuration to display, the default value is used.

Parameter	Default value	Description
title	document.title	The title to display in the searchahead embed.
image	document.thumbnailImage	The thumbnail image to display in the searchahead embed.

Configuration example

The information passed in an HTML page must use valid JSON syntax that can be parsed, converted, and applied.

```
<lw-searchahead placeholder="Search the documentation" type-
debounce="20" config='{ "limit": 3, "highlight": false, "filters":
[{"field": "file.extension", "values": ["pdf"]}], "sort":
[{"sortField": "document.sourceLastModifiedTimestamp", "sortOrder":
"desc"}], "fieldList": ["CUSTOM_TITLE_METATAG",
"CUSTOM_IMAGE_METATAG"], "displayFields": { "title":
"CUSTOM_TITLE_METATAG", "image": "CUSTOM_IMAGE_METATAG" },
"analyticsData": false}'></lw-searchahead>
```



You can also send valid JavaScript object syntax. For more information about JSON formatting, see [W3Schools JSON Syntax](#) .




CSS styling

Searchahead embed

Reference

The searchahead embed includes CSS custom properties so you can match the searchahead embed's appearance to your website's theme and branding.

Consult the following resources for general CSS guidance and tutorials:

- [W3Schools CSS tutorial](#) 
- [MDN CSS reference](#) 
- [MDN CSS custom properties](#) 

The following code sample displays the searchahead embed's CSS custom variables and their default values. Click **Edit on CodePen** to open a new screen and save your work.

HTML

CSS


Result

EDIT ON

LIVE

```
lw-searchahead {
  /* Input CSS variables
   Input variables control the
   appearance of the box that the
   user types the search query
   into.*/
  --font-size: 12px;
  --input-width: 640px;
  --input-height: 32px;
  --input-font-size: var(--font-
size);
  --input-padding: 4px;
  --input-margin: 0;
  --input-box-shadow: none;
  --input-border: 1px solid
#ebeced;
  --input-border-radius: 4px;
  --input-focus-border-color:
#666;
  --input-hover-border-color:
#999;
  --input-focus-box-shadow:
none;
  --input-bg-color: #f9fafb;
  --input-font-color: #0b1731;
  --input-flex-direction: row;
  --input-focus-outline: none;
  --input-cursor: default;
  --input-overflow: hidden;

  /* Search icon variables
   Search icon variables control
   the placement and appearance of
   the search icon used to complete
```



Resources1x0.5x0.25xRerun

Search results

Embed configuration options

Reference

The search results embed displays the result returned by your application. Result cards include the title, description, and URL of the result. Sort options include relevance, modification date, and publication date.

The search results embed also collects *click signals* when an end user clicks a document. The embeds send these click signals to Springboard. No additional setup is required to collect signals.

For conceptual information, see *Signals*.

```
<div> <lw-results></lw-results> </div>
```

Result



Tip

Try it!

To test the embed, complete the following:

- Click a result.
- Click through the pages of results.
- Sort the results.
- Choose how many results to display.



Note

The preceding example limits the number of search results returned using the `limit` configuration option. The default number is 20.


Configuration

Search results embed

Reference

The search results embed offers configuration options to customize what displays in your embed.

Configuration options

Parameter	Values	Description
<code>resultCount</code>	number 5 50	Controls the maximum number of results returned by the object. This parameter is outside of the <code>config</code> string. The <i>configuration example</i> displays the appropriate syntax.
<code>limit</code>	number	The maximum number of results returned per page. This does not affect the total number of documents in the result set, or "hits." <ul style="list-style-type: none"> Default value: <code>20</code>
<code>highlight</code>	boolean	Enables or disables highlighting of results. <ul style="list-style-type: none"> Default value: <code>true</code>
<code>filters</code>	array	Specifies which facets to select. The format is an array of objects where each object contains a <code>field</code> and a <code>value</code> .
<code>sort</code>	array	Specifies sort order. <ul style="list-style-type: none"> Default value: <code>relevancy asc</code>
<code>displayFields</code>	object	Specifies the metatags to use when displaying the fields that display in the search results embed. If you do not specify a metadata tag in the searchahead configuration to display, the default value is used.
<code>deepLinking</code>	boolean or object	Customizes the deep linking options in the embed. You can specify the URL query parameters for the embed to use. For example, if you use <code>s</code> to indicate the sort parameter elsewhere on your website, Springboard can continue using <code>s</code> as your sort parameter instead of <code>sort</code> . Springboard can customize the <code>page</code> , <code>sort</code> , and <code>order</code> field names. <ul style="list-style-type: none"> If <code>"deepLinking": false</code> in the embed configuration, then deep linking is not enabled for that embed, and any parameters set in that embed are not included in the URL. If <code>deep-linking = false</code> or does not exist in the <i>embed configuration script</i>, adding values to the <code>deepLinking</code> field in the embed configuration has no effect. <div>  Note The <code>page</code> value starts at <code>0</code> to match the API. </div>
<code>group</code>	string	Specifies the group that your embeds are in. A group of embeds uses the same deep linking and URL query parameter values. This parameter is outside of the <code>config</code> value. For example, using the search results embed in group A does not affect the embeds in group B, even if group B includes another search results embed.

Parameter	Values	Description
<code>analyticsData</code>	boolean	<p>If true, the request is included in analytical data. Set <code>analyticsData</code> to false to exclude the query from the application metrics and other analytical data. Set to false for all non-production requests, including tests.</p> <p>This field does not affect the query or results.</p> <ul style="list-style-type: none"> Default value: <code>true</code>

Display fields customization

You can use the contents of a different metadata tag to override the fields that display in the search results embed.

If you use a metadata tag for one of the display fields, that metadata field must be indexed in *your data source*. If you specify a metadata tag in the configuration options and the metadata tag is not available in the document, then Springboard uses the default value for that parameter.

If you do not specify a metadata tag in the results configuration to display, the default value is used.

Parameter	Default value	Description
<code>title</code>	<code>document.title</code>	The title to display in the search results.
<code>description</code>	<code>document.description</code>	The description to display in the search results.
<code>url</code>	<code>document.url</code>	The URL to display in the search results.
<code>image</code>	<code>document.thumbnailImage</code>	The thumbnail image to display in the search results.

Configuration example

The information passed in an HTML page must use valid JSON syntax that can be parsed, converted, and applied.

```
<lw-results resultCount="50" config='{ "limit": 3, "highlight": false,
"filters": [{"field": "file.extension", "values": ["pdf"]}], "sort":
[{"sortField": "document.sourceLastModifiedTimestamp", "sortOrder":
"desc"}], "displayFields": { "title": "YOUR_TITLE_METATAG", "url":
"YOUR_URL_METATAG" }, "analyticsData": false}'></lw-results>
```

Note

You can also send valid JavaScript object syntax. For more information about JSON formatting, see [W3Schools JSON Syntax](#).




CSS styling

Search results embed

Reference

The search results embed includes CSS custom properties so you can match the search results embed's appearance to your website's theme and branding.

Consult the following resources for general CSS guidance and tutorials:

- [W3Schools CSS tutorial](#) 
- [MDN CSS reference](#) 
- [MDN CSS custom properties](#) 

The following code sample displays the search results embed's CSS custom variables and their default values. Click **Edit on CodePen** to open a new screen and save your work.

HTML

CSS

Result


EDIT ON

LIVE

```
lw-results {
  /* Results box CSS
   The results variables control
   the appearance of the results
   box.*/
  --results-border: solid 1px
#EBECED;
  --results-border-radius:
4px;
  --results-padding: 0 16px;

  /* Title CSS
   Results title variables
   control the appearance of the
   "Results List" title.*/
  --results-title-display:
flex;
  --results-title-font-size:
16px;
  --results-title-color:
#333333;

  /* Results list CSS
   Results list variables control
   the appearance of the full
   results list.*/
  --results-list-overflow:
hidden;
  --results-list-padding-top:
44px;
  --results-list-border-
radius: 4px;
  --results-list-bg-color:
#FFFFFF;
```



Resources1x 0.5x 0.25xRerun

Search Store

Concept

The Search Store API client allows you to integrate your application with your search interface through token authentication, instead of developing your backend to use OAuth 2.0.

The Search Store makes use of the *token utilized by embeds*. To enable embeds and retrieve an embed token:

1. Click the Integrations icon from the *Applications UI sidebar*.
2. From the Embeds tab, click the toggle to make your application **Public**.
3. Add your domains to the **Allowed origins** field.
4. Click **Copy** to copy the embed script and style sheet. Paste it in the `<head>` of your HTML file.

```
<script type="module" id="lw-ui-lib" api-  
url="https://APPLICATION_ID.applications.lucidworks.com/" embed-  
token="EMBED_TOKEN" src="SCRIPT_SRC"></script>
```

? FAQ

Can I use embed tokens to connect to multiple Springboard applications on the same site?

No, your site can only connect to one *Springboard application* at a time with embed tokens.

This also applies to subdomains of your site. If your site, `example.com`, uses embeds with your Springboard application, you can't connect `internal.example.com` to a different Springboard application using embed tokens.

Additional information

The *Developer documentation* is a useful resource for writing Search Store functions, as it includes API configuration specifications with full descriptions and allowed values for the Search Store functions.

API functions

Search Store API client

Concept

API functions allow you to pass a value directly to an endpoint to receive the JSON response generated by a comparable Query API or Signals API call. Each endpoint has a corresponding stateless API function.

Function value formats

Depending on the function used, you can choose to pass a string or *JSON object* API parameters. For example, you can pass the query `books` as a string to the `search` function:

```
window.getSearchStore().search('books')
```

You can also pass the query as a JSON object:

```
window.getSearchStore().search({query: 'books'})
```

By using a JSON object, you can make your request more specific. For example, you can specify that you want the third page of results for the query:

```
window.getSearchStore().search({ query: 'books', page: 3 })
```

Examples

Required parameters

You can pass the query `books` as a string to the `search` function to receive the results JSON response:

Request

```
window.getSearchStore().searchahead('books')
```

Response

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "hits" : 2,
  "docs" : [ { "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 3,
    "datasourceLabels" : [ "books" ], "fields" : { "document.title" :
      "Garfield out to lunch / by Jim Davis.", "document.url" :
        "https://www.example.com/docs/seattle-books-meta/2375907.html" } }, {
      "type" : "page", "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731", "rank" : 2,
      "datasourceLabels" : [ "books" ], "fields" : { "document.title" :
        "Percy Jackson & the Olympians. Book two, The sea of monsters : the
        graphic novel / by Rick Riordan ;...", "document.url" :
          "https://www.example.com/docs/seattle-books-meta/2921533.html" } } ] }
```

Similarly, you can use the **detail** function to pass a document ID value to retrieve the details for that document:

Request

```
window.getSearchStore().detail ('441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731')
```

Response

```
{ "queryId" : "fd110486-f168-47c0-a419-1518a4840589", "type" : "page",
  "id" : "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731",
  "datasourceLabels" : [ "books" ], "fields" : { "document.title" :
    "Garfield out to lunch / by Jim Davis.", "document.body" : "Garfield
    Fictitious character Comic books strips etc, Cats Comic books strips
    etc, Cartoons and comi... Garfield out to lunch / by Jim Davis. Info
    author pubYear publisher bibNumber Davis, Jim, 1945 July 28- 2006.
    Ballantine Books, 2375907", "document.url" :
    "https://www.example.com/docs/seattle-books-meta/2375907.html",
    "document.sourceCreateTimestamp" : "2022-04-21T15:39:45Z",
    "document.sourceLastModifiedTimestamp" : "2022-04-21T15:39:45Z",
    "document.languageCode" : "en" } }
```

If you want to send a click signal, which helps improve relevancy over time, use the **clickSignal** function. Use the same document ID for both functions. The example below also includes the optional parameter **queryId** in the **clickSignal** function.

Request

```
window.getSearchStore().clickSignal({documentId: '441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731', queryId: 'fd110486-f168-47c0-a419-1518a4840589'}) window.getSearchStore().detail('441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-68708807657148371333355819934570439731')
```

Response

No response is given for the **clickSignal** function. Instead, your application receives a click signal associated to your query and click action.

Required and optional parameters

You can also pass a JSON object, to use additional API parameters and values with the function. For example, to send the query **books** with the **search** function while specifying the results page number and facets returned:

Request

```
window.getSearchStore().search({ query: 'books', page: 3, facets: [{  
  field: 'type', values: ['pdf', 'page'] }], analyticsData: true })
```

Search

Search Store API functions

Reference

The **search** endpoint of the Query API searches an indexed collection of data for relevant full-term matches to user-specified search terms. It powers the search feature in your search application.

Pass a value as a string. Alternatively, pass a JSON object to include optional parameters.

Example requests

Required parameters

```
window.getSearchStore().search('QUERY')
```

Required and optional parameters

```
window.getSearchStore().search({ query: 'QUERY', page: PAGE, limit:
LIMIT, sort: [{ sortField: 'SORT_FIELD', sortOrder: 'SORT_ORDER' }],
facets: ['FACET_1', 'FACET_2'], fieldList: ['FIELD_1', 'FIELD_2'],
highlight: HIGHLIGHT_BOOLEAN, filters: [{ field: 'FIELD', values:
['VALUE_1', 'VALUE_2'] }], analyticsData: true })
```

Important

The preceding example demonstrates the required parameters and some optional parameters to pass with this function. For full details on all parameters and allowed values, see the [Query API configuration spec](#).

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.

Tip

Edit the function and click the corresponding button again to see the result change.


HTMLJSResultEDIT ON

LIVE

```
// Retrieve search store early
in the lifecycle. Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Search
async function searchBasic() {
  console.log("Basic search
triggered:");
  return
window.getSearchStore().search("
book").then(console.log);
}

async function searchAdvanced()
{
  console.log("Advanced search
triggered:");
  return window
    .getSearchStore()
    .search({
      query: "book",
      page: 1,
      limit: 2,
      sort: [
        {
          sortField:
"relevancy",
          sortOrder: "asc"
        }
      ],
      facets: ["type",
"document.languageCode"],
```



Resources1x0.5x0.25xRerun

Searchahead

Search Store API functions

Reference

The **searchahead** endpoint of the Query API searches a collection of data for relevant partial-term matches to user-specified search terms. It powers the searchahead feature in your search application's UI, which displays real-time query matches as the user types. The query matches improve as the user continues typing.

Pass a value as a string. The value must be three characters or more in length. Alternatively, pass a JSON object to include optional parameters.

Example requests

Required parameters

```
window.getSearchStore().searchahead('QUERY')
```

Required and optional parameters

```
window.getSearchStore().searchahead({ query: 'QUERY', page: PAGE,  
  limit: LIMIT, sort: [{ sortField: 'SORT_FIELD', sortOrder:  
    'SORT_ORDER' }], fieldList: ['FIELD_1', 'FIELD_2'], highlight:  
  HIGHLIGHT_BOOLEAN, filters: [{ field: 'FIELD', values: ['VALUE_1',  
    'VALUE_2'] }], analyticsData: true })
```

Important

The preceding example demonstrates the required parameters and some optional parameters to pass with this function. For full details on all parameters and allowed values, see the *Query API configuration spec*.

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.

Tip

Edit the function and click the corresponding button again to see the result change.

HTML

JS

Result

EDIT ON

LIVE

```
// Retrieve the Search Store
early in the lifecycle.
Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Searchahead
async function
searchaheadBasic() {
  console.log("Basic searchahead
triggered:");
  return
window.window.getSearchStore().s
earchahead("book").then(console.
log);
}

async function
searchaheadAdvanced() {
  console.log("Advanced
searchahead triggered:");
  return window.window
    .getSearchStore()
    .searchahead({
      query: "book",
      page: 1,
      limit: 2,
      sort: [
        {
          sortField:
"relevancy",
          sortOrder: "asc"
        }
      ]
    })
}
```



Resources

1x 0.5x 0.25x

Rerun

Detail

Search Store API functions

Reference

The **detail** endpoint of the Query API retrieves metadata associated with a specific record.

Pass a document ID value as a string. This value is returned by the **search**, **searchahead**, and **browse** functions at **docs.id**. Alternatively, pass a JSON object to include optional parameters.

Example requests

Required parameters

```
window.getSearchStore().detail('DOC_ID')
```

Required and optional parameters

```
window.getSearchStore().detail({ id: 'DOC_ID', fieldList: ['FIELD_1',  
'FIELD_2'], analyticsData: true })
```

Important

The preceding example demonstrates the required parameters and some optional parameters to pass with this function. For full details on all parameters and allowed values, see the *Query API configuration spec*.

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.

Tip

Edit the function and click the corresponding button again to see the result change.

HTMLJSResultEDIT ON

LIVE

```
// Retrieve the Search Store
early in the lifecycle.
Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Detail
async function detailBasic() {
  console.log("Basic detail
triggered:");
  return window
    .getSearchStore()
    .detail(
      "2ea44bf2-14d6-4c33-a1d6-
7ff6e5fc45dc-7ca9b565-07c4-4b8e-
a770-12f4749daded-
73717843844562742277336285184299
742787"
    )
    .then(console.log);
}

async function detailAdvanced()
{
  console.log("Advanced detail
triggered:");
  return window
    .getSearchStore()
    .detail({
      id:
        "2ea44bf2-14d6-4c33-
a1d6-7ff6e5fc45dc-7ca9b565-07c4-
4b8e-a770-12f4749daded-
```



Resources1x0.5x0.25xRerun

Browse

Search Store API functions

Reference

The **browse** endpoint enables developers to return all documents in the current application, sorted by relevancy. Additional sort and filter options are available.

This function doesn't require a value. Alternatively, pass a JSON object to include optional parameters.

Example requests

Required parameters

```
window.getSearchStore().browse()
```

Required and optional parameters

```
window.getSearchStore().browse({ page: PAGE, limit: LIMIT, sort: [{  
  sortField: 'SORT_FIELD', sortOrder: 'SORT_ORDER' }], facets:  
  ['FACET_1', 'FACET_2'], fieldList: ['FIELD_1', 'FIELD_2'], filters: [{  
    field: 'FIELD', values: ['VALUE_1', 'VALUE_2'] }], analyticsData: true  
})
```

Important

The preceding example demonstrates the required parameters and some optional parameters to pass with this function. For full details on all parameters and allowed values, see the *Query API configuration spec*.

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.

Tip

Edit the function and click the corresponding button again to see the result change.

HTML

JS

Result

EDIT ON

LIVE

```
// Retrieve the Search Store
early in the lifecycle.
Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Browse
async function browseBasic() {
  console.log("Basic browse
triggered:");
  return
window.getSearchStore().browse()
.then(console.log);
}

async function browseAdvanced()
{
  console.log("Advanced browse
triggered:");
  return window
    .getSearchStore()
    .browse({
      page: 1,
      limit: 2,
      sort: [
        {
          sortField:
"relevancy",
          sortOrder: "asc"
        }
      ],
      facets: ["type",
"document.languageCode"],
```



Resources1x0.5x0.25xRerun

Metatags

Search Store API functions

Reference

The `metatags` endpoint of the Query API retrieves all the *custom metadata tags* associated with an application, where at least one document contains data in that metatag.

This function doesn't accept a value.

Example requests

```
window.getSearchStore().metatags()
```

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.



Tip

Edit the function and click the corresponding button again to see the result change.

HTML

JS


Result

EDIT ON

LIVE

```
// Retrieve the Search Store
early in the lifecycle.
Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Metatags
async function metatagsBasic() {
  console.log("Basic metatags
triggered:");
  return
window.getSearchStore().metatags(
).then(console.log);
}
```



Resources

1x 0.5x 0.25x

Rerun

Signals

Search Store API functions

Reference

Click signals

Pass a value as a JSON object that includes, at a minimum, the document ID. This value is returned by the `search`, `searchahead`, and `browse` functions at `docs.id`. You can include optional parameters in the JSON object.

The following table describes the required parameters and some recommended optional parameters to pass with this function. For full details on all parameters and allowed values, see the *Signals API configuration specifications*. For conceptual information, see *Signals*.

Parameter	Type	Description
Required		
<code>documentId</code>	string	The unique identifier for the record within the indexed collection.
Optional		
<code>queryId</code>	string	The unique identifier for the query.
<code>timestamp</code>	integer	The date and time the signal was recorded. Format this value as Unix epoch time, in milliseconds. If this optional parameter isn't used, the current time is passed by default.

Example requests

Required parameters

```
window.getSearchStore().clickSignal({documentId: 'DOC_ID'})
```

Required and optional parameters

```
window.getSearchStore().clickSignal({ documentId: 'DOC_ID', queryId: 'QUERY_ID', timestamp: TIMESTAMP })
```

Demo

The following CodePen resource demonstrates the usage of this function. Open your browser's web console and network monitor logs and watch for activity after clicking the buttons that trigger the Search Store API function.



Edit the function and click the corresponding button again to see the result change.

HTML

JS

Result

EDIT ON

LIVE

```
// Retrieve the Search Store
early in the lifecycle.
Otherwise, use
`window.getSearchStore()`.
window.onload = async () => {};

// API functions
// Signals
async function signalsBasic() {
  console.log("Basic signals
triggered:");
  return window
    .getSearchStore()
    .clickSignal({
      documentId:
        "2ea44bf2-14d6-4c33-
a1d6-7ff6e5fc45dc-7ca9b565-07c4-
4b8e-a770-12f4749daded-
73717843844562742277336285184299
742787"
    })
    .then(console.log);
}

async function signalsAdvanced()
{
  console.log("Advanced signals
triggered:");
  return window
    .getSearchStore()
    .clickSignal({
      documentId:
        "2ea44bf2-14d6-4c33-
a1d6-7ff6e5fc45dc-7ca9b565-07c4-
```

Resources1x0.5x0.25xRerun

Query signals

Pass a value as a JSON object that includes, at a minimum, the search terms and document ID, The query ID is optional.

The following table describes the parameters to pass with this function. For full details on all parameters and allowed values, see the *Signals API configuration specifications*. For conceptual information, see *Signals*.

Parameter	Type	Description
Required		
searchTerms	string	The term or phrase you are searching.
documentIds	array	The unique identifiers for the documents submitted in the query.
Optional		
queryId	string	The unique identifier for the query.

Example requests

Required parameters

```
window.getSearchStore().querySignal({ searchTerms: 'SEARCH_TERMS',
documentIds: [ "DOC_ID1", "DOC_ID2", "DOC_ID3" ] })
```

Required and optional parameters

```
window.getSearchStore().querySignal({ searchTerms: 'SEARCH_TERMS',
documentIds: [ "DOC_ID1", "DOC_ID2", "DOC_ID3" ], queryId: 'QUERY_ID'
})
```

Observable and stateful functions

Search Store API client

Reference

Observable functions, such as `subscribe`, accept callback functions — sometimes referred to as subscribers — that are called when the Search Store is updated.

The Search Store is updated by stateful functions, such as `setQuery`, which retrieve data from a Query API endpoint.

The table below details each function, including a description of its usage.

Function	Usage
Observable	
<code>window.getSearchStore().subscribe(OBJECT)</code>	Subscribe to changes to stateful functions. Replace OBJECT with the name you assign to the response object.
Stateful	
<code>window.getSearchStore().setQuery('QUERY')</code>	Pass a string to set the query.
<code>window.getSearchStore().setFacets([{'field': 'FIELD', 'values': ['VALUE_1', 'VALUE_2']}])</code>	Pass a JSON object to set the facets returned in the response.
<code>window.getSearchStore().setPage(PAGE_NUMBER)</code>	Pass an integer to set the page number of results to return. If the page number value is greater than the actual number of pages of results, the response is an empty docs array.
<code>window.getSearchStore().setSort('FIELD', 'VALUE')</code>	Pass two comma-separated strings to set the sort value.
<code>window.getSearchStore().setSearch(JSON_OBJECT)</code>	Pass a JSON object that includes, at a minimum, a query.



Tip

The *Query API configuration spec* includes full descriptions of the parameters used by stateful functions, including the allowable values.

Examples

Basic

Use `setQuery` to set the query to `books` and use the query in a `searchahead` or `search` function. As an example use case, you can pass the query to the `searchahead` function right away while storing the value for the `search` function until the user presses **Enter** to submit the query.

```
window.getSearchStore().setQuery('books')
```

Similarly, you can use **setPage** to set the results page value to **2** and use the value while performing a search or browse operation.

```
window.getSearchStore().setPage(2)
```

Advanced

Use **setSearch** to set the values for multiple search fields using a JSON object. See the *Query API configuration spec* for full descriptions of the parameters, including the allowable values.

```
window.getSearchStore().setSearch({ "query": "books", "page": 1,  
  "sort": [{ "sortField": "relevancy", "sortOrder": "asc" }],  
  "fieldList": [ "document.title", "document.url" ], "highlight": true  
})
```

Then, use other observable functions to override the values set by **setSearch**. For example, you can change the query to **movies** using the **setQuery** function and the page value to **5** using the **setPage** function. The request JSON sent uses these values for the query and page number but still uses the other fields set by **setSearch**.

Functions

```
window.getSearchStore().setQuery('movies');  
window.getSearchStore().setPage(5);
```

The full JSON request object includes the updated values.

```
{ "query": "movies", "highlight": true, "facets": ["file.extension",
"document.languageCode", "datasourceLabels", "type"], "utcOffset":
"-06:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "sort":
[{ "sortField": "relevancy", "sortOrder": "asc" }], "page": 5 }
```

Subscribe

Use the **subscribe** function to define actions to take when a stateful function makes a change. The example below demonstrates how to update the page number with a button.

```
<script> window.onload = async () => {
window.getSearchStore().subscribe((searchStoreResponse) => {
console.log('Your results:', searchStoreResponse); 3 });
window.getSearchStore().setSearch({ "query": "books", "page": 1, 2
"sort": [{ "sortField": "relevancy", "sortOrder": "asc" }],
"fieldList": [ "document.title", "document.url" ], "highlight": true
}); } </script> <button
onclick="window.getSearchStore().setPage(2)">Page 2</button> 1
```

1	When a user clicks the button, the setPage function sets the page value to 2 .
2	This overrides the page value 1 in the setSearch function.
3	The subscribe function is triggered, and the response is logged to the console.

Signals

Concept

Signals provide insights into the terms and phrases users enter to search for information and the results that most closely match the queries.

Springboard supports:

- **Click signals.** When a user clicks to perform any action in the application, a signal is recorded. One example of a click signal is when a user clicks a search result, which is a strong indicator the result is relevant.
- **Query signals.** When a user clicks to send a query, a signal is recorded.

To help improve relevancy over time, you can share signals with Springboard using the:

- *Signals API*
- *Search Store API functions*
- *Search results embed*

Example scenario

An example of how signals can improve relevancy is as follows:

1. The user searches for **children's books**.
2. A query signal is recorded.
3. The user clicks the third result, which is **100 Best Children's Books of the 90's**.
4. The click signal is recorded.
5. More users send the same query for **children's books** and click the same result.
6. Each query and click signal is recorded.
7. If there are more click signals for that result than any other, then Springboard determines **100 Best Children's Books of the 90's** should be the first result for the **children's books** query.

In summary, although Springboard initially determined **100 Best Children's Books of the 90's** should be the third result based on the content's relevancy, click signals proved users were more interested in that result than the others so the relevancy improved.

Regions

Concept

Regions are physical locations around the world where your Springboard data is stored, queried, and ingested. Springboard stores your applications and data sources in regions and lets you choose a region when creating an application or data source.

The region that your application is stored in determines where your application data is stored and queried.

The region that your data source is stored in determines where your data source information is ingested.



Caution

Region selection is permanent.

Region selection guidelines

Typically, you store your application data and data source information in the same region, as this creates the best experience for your users.

You can select a region that is physically close to your users. This improves latency so users receive their results faster.

You can also select a region whose data storage and processing requirements you need to adhere to. If this is the case, you must select the same region for your applications and its associated data sources.

You may store your applications and associated data sources in different regions. For example, you may process data in one region even though your users query the data from another region. In this situation, selecting one region for your data sources and the region closest to your userbase improves latency at the query and ingestion stages.

Available regions

The values for the region field are `eu-london` and `us-southcarolina`.

Relevancy controls

Concept

Relevancy controls allow for fine-tuning of query results. You can override the default search results for individual queries that your customers make. These adjustments help deliver relevant results to customers when the system's default relevance is not adequate. This feature accomplishes goals such as removing irrelevant or expired documents from search results and moving certain documents to the top of the results list.

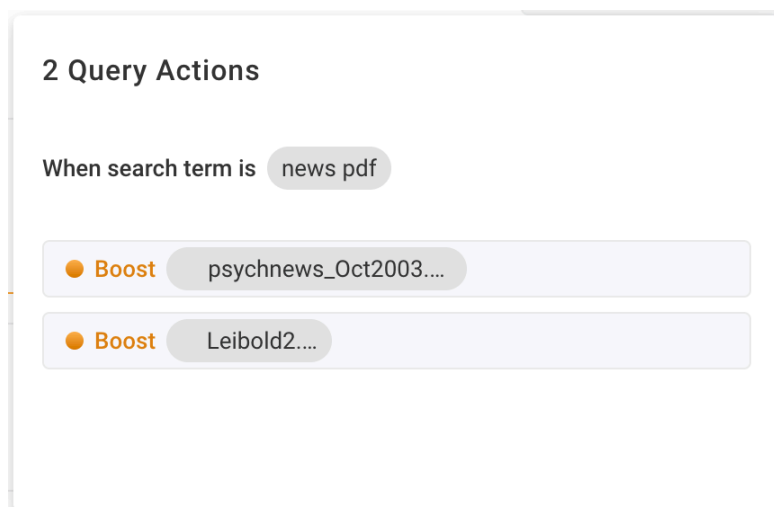
Boost

Boost artificially increases the relevancy rank to ensure users see a specified result. You can boost a document so that it occupies a position at the top of the search results when your customers run a particular query. Because boosting is tied to relevancy rank, this action only applies when documents are sorted by relevance from highest to lowest rank.

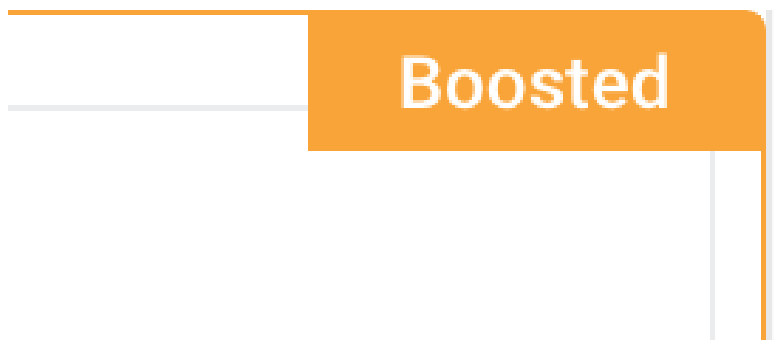
Boosting is an action you can set in the UI. Perform a query using the search term you want to manipulate. When you point to a document in the search results, an option to boost appears.



You can then select the **Boost** button to make that document appear higher in future results for that query. This button associates a boost action for the document with the search term.



Boosted documents appear at the top of the results surrounded by a border.



Use the Query Actions button to review actions associated with the current query.

● 2 Query Actions

Remove the boosted item from the list of actions when you no longer want the document to show up higher in search results. The relevancy rank for the document is then calculated as normal on subsequent queries.

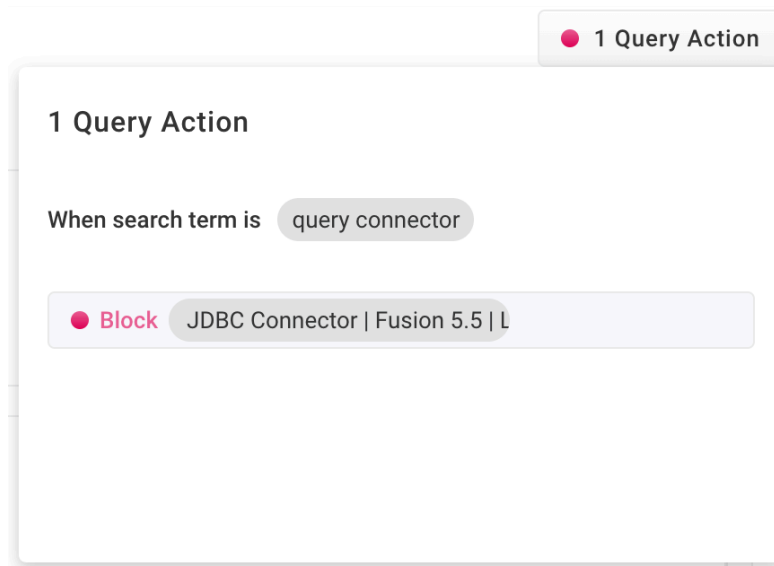
Block

Block prevents documents from showing in results if they are irrelevant, out of date, or should otherwise not be included.

Blocking is an action you can set in the UI. When you point to a document in search results, an option to block appears.



You can then select the **Block** button to add it to the list of query actions and prevent that document from appearing in future search results.



Remove the blocked item from the list of actions when you want to add the document back into search results. If you want to block a document that is boosted, remove the rule for the boost from that document in the list of actions and the option to block appears.

Queries with actions

Use the Action Summary button to show a list of modified queries.

☰ Action Summary

For example, in Experience Optimizer, you can see the list of all query terms that have actions applied to them.

All Modified Search Terms

To view and edit actions associated with a search term, select the term and click View.

Search

connection
pdf
weekly

Cancel

View

Search is enabled to navigate quickly, which is helpful if you happen to have a large number of modified queries.

Query rule limits

The following limits apply to query rules:


- You can apply rules to a maximum of 200 queries
- For each individual query, you can apply a maximum of 50 rules

The absolute maximum number of rules that can be made for an application is 10,000 (200 queries with 50 rules each).

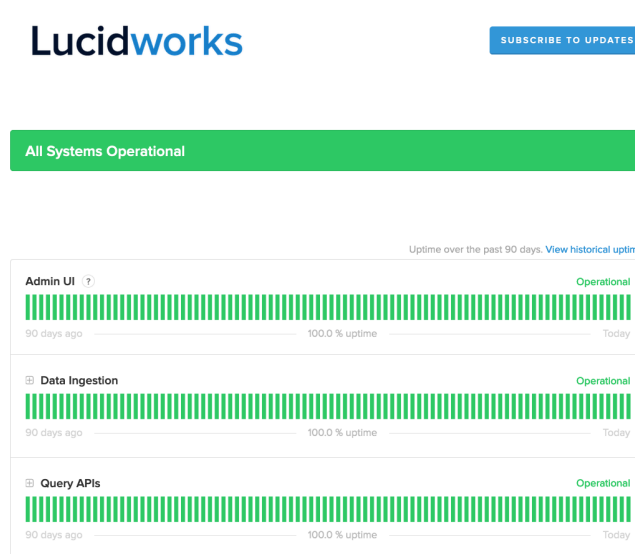
Support

Concept

Status page

If you're encountering unexpected behavior in Springboard, review the [Lucidworks status page](#) . When a Springboard service is facing a decrease in performance or an outage, it's reported here as an "incident."

An incident description includes impacted service, the effect this has on users, and the current resolution status.



If the status page doesn't explain the issue you're having, consider *contacting support*.

Contact support

If you would like Lucidworks support to investigate an issue you're having, you can submit a ticket from Springboard.

1. From any screen within Springboard, click the support icon.



In the **Need Help?** screen, the values of the user signed in to Springboard automatically display in the **Your name** and **Email address** fields.

2. Complete the form.
3. Click **Send** to submit your support request for review or click **X** to exit without sending the request.

FAQ

What if I can't contact support in Springboard?

If you aren't able to contact support from within Springboard, you can contact support directly at support@lucidworks.com.

Documentation sign in help

Prerequisite

You must be a Springboard customer to view Springboard documentation. If you are not signed in to Springboard and you attempt to view a Springboard topic, you will be prompted to sign in.

Note

Your session is synced between Springboard and the documentation. If you are prompted to sign in to view the documentation despite already being signed into Springboard, ensure your browser is not blocking third-party cookies.

Sign in

Important

Use the same credentials you use to sign in to Springboard.

Enter your Springboard **Username** and **Password** and click **Sign in**. To retain information when you open the application for future sessions, select the **Remember me** check box.

Help signing in

If you are unable to sign in, contact support at support@lucidworks.com.