

LucidWorks Search REST API Reference Guide

2.7 Documentation

Table of Contents

How to Use this Documentation	5
Audience and Scope	5
Conventions	5
Customers of LucidWorks Search on AWS or Azure	7
Getting Support & Training	8
LucidWorks REST API Reference	9
About the LucidWorks Search REST APIs	9
Quick Start	10
List of Available APIs	10
Getting Started Indexing	13
Error Response Format	17
Overview of REST API calls	19
Pretty Printing JSON Output	25
Version	26
Collections	29
Alerts API	685
Users	696
SSL Configuration	703
Crawler Status	706
Example Clients	710
Advanced Operations Using the REST API	727
About LucidWorks	731

LucidWorks Search Documentation

The LucidWorks Search Documentation is organized into several guides that cover all aspects of using and implementing a search application with LucidWorks Search, whether on-premise or hosted on AWS or Azure.

Installation & Upgrade Guide

- [Installing LucidWorks Search](#)
- [System Directories and Logs](#)
- [Upgrade instructions for v2.6](#)
- Review [changes from LucidWorks v2.5 to v2.6](#)

System Configuration Guide

- [Troubleshooting crawl issues](#)
- [Alerts configuration](#)
- [Query options](#)
- Custom [fields](#), field types, and other [index customizations](#)
- [Performance considerations](#) and [system monitoring](#)
- [Distributed search and indexing](#)
- [Security options](#)

Lucid Query Parser

- [How the default query parser handles user requests](#)
- [Customization options](#)

LucidWorks REST API Reference

- Configure [data sources](#) and [administer crawls](#)
- [Set system settings](#)
- Manage [fields](#), [field types](#), and [collections](#)
- Example clients in [C#](#), [Perl](#) and [Python](#)

Custom Connector Guide

- [Introduction to Lucid Connector Framework](#)
- [How To Create A Connector](#)

How to Use this Documentation

Audience and Scope

This guide is intended for search application developers and administrators who want to use LucidWorks Search to create world class search applications for their websites.

While LucidWorks Search is built on Solr, and many of its features are implementations of Solr and Lucene features, this Guide does not cover basic Solr or Lucene configuration. We do, however, point out where LucidWorks Search deviates from Solr or Lucene standard configuration practices, and have provided links to Solr and Lucene documentation where possible for further explanation if the functionality in LucidWorks Search is identical to Solr or Lucene.

One important note to remember is that LucidWorks is multi-core enabled by default, with `collection1` as the default core. This means that standard Solr paths such as `http://localhost:port/solr/*`, as shown in Solr documentation, would be `http://localhost:port/solr/collection1/*` in LucidWorks Search.

Topics covered on this page:

- [Audience and Scope](#)
- [Conventions](#)
- [Customers of LucidWorks Search on AWS or Azure](#)
- [Getting Support & Training](#)

Conventions






Paths

Server paths are described in relation to the base LucidWorks Search installation path, indicated by `$LWS_HOME`. For example, if LucidWorks Search was installed at `/var/lucidworks`, then the path to the 'app' directory shown as `$LWS_HOME/app` will be `/var/lucidworks/app` on the server.

Notes

Special notes are included throughout these pages.

Note Type	Look & Description
-----------	--------------------

Note Type	Look & Description
Information	 Notes with a blue background are used for information that is important for you to know.
Notes	 Notes are further clarifications of important points to keep in mind while using LucidWorks.
Tip	 Notes with a green background are Helpful Tips.
Warning	 Notes with a red background are warning messages.
Cloud	 Information for LucidWorks Search in the Cloud Users Information specifically for LucidWorks Search customers on the AWS or Azure Platform.

REST API Conventions

Many of the LucidWorks Search [REST APIs](#) support several methods (such as POST, GET, PUT, DELETE) and each is documented with detailed attribute descriptions and examples of inputs and outputs. Each description includes the path to the API endpoint, parameters for input, and the attributes returned as a result of the request.

Windows users should take care when copying the examples as they assume that you are familiar with how to modify unix-based curl commands for the Windows environment.

Parameters

Several of the paths shown in the API documentation include parameters that need to be modified for your installation and specific configuration. These are indicated in *italics*.

For example, getting the details of a data source is shown as:

```
GET /api/collection/collection/datasources/id.
```

If you were using 'collection1' and data source '3', you would enter:

```
GET /api/collection/collection1/datasources/3.
```

Server Addresses

The LucidWorks Search REST API uses the [Core component](#), installed at <http://localhost:8888/> by default in LucidWorks Search. Many examples in this Guide use this as the server location. If you have installed LucidWorks Search locally, and you changed this location on install, be sure to change the destination of your API requests accordingly.

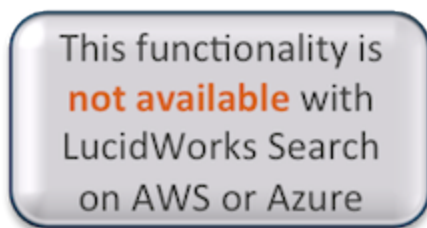
Customers hosted on AWS or Azure should see the section for [#Customers of LucidWorks Search on AWS or Azure](#) below.

Customers of LucidWorks Search on AWS or Azure

All of the preceding information on this page applies to customers who have LucidWorks Search hosted on either AWS or Azure Platforms, with a few small exception which are detailed below.

Configuration Options

Certain configuration options are available with on-premise installations only (such as installation options, manual configuration file changes, etc.). The following panel will appear on any page or section that does not apply or is not available for LucidWorks Search on the AWS or Azure platforms:



API Conventions for LucidWorks Search on AWS or Azure

Nearly all of the documented REST APIs will work for customers on AWS or Azure, but the example API calls must be modified to include either the Access Key or the API Key and used as authentication credentials. Customers are being transitioned from a simple Access Key to a more secure Basic authentication system that requires a unique API Key.

1. Customers who only have an Access Key can see the key on the My Search Server page and the main Collections Overview page of your instance (click the REST API button above the usage graphs). Example URLs for API calls used in this documentation would then be changed from `http://localhost:8989/api/...` to `http://access.lucidworks.io/<access key>/api/...`. This access key is specific to your instance and should be treated as securely as possible to prevent unauthorized access via the APIs to your system.

2. Customers with Basic authentication have instances which use an URL with "https://s-**XXXXXXXX**.lucidworks.io" where **XXXXXXXX** is 8 characters (letters or numbers). So, if your instance URL is "https://s-9sdf10b.lucidworks.io/" you would use that in place of any example API calls that used "http://localhost:8888". For example, this call to get all collections:

```
curl 'http://localhost:8888/api/collections'
```

would be changed to:

```
curl -u 'API_Key:password' 'https://s-9sdf10b.lucidworks.io/api/collections'
```

The API_Key can be found by logging in to your LucidWorks Search instance, and clicking "My Account" at the upper right of the screen. Click "API Access" on the left to view the API key. The password is 'x' by default. There is not currently a way to change the default password. You should take care not to expose this key when posting to our forums, as that information could be seen by other LucidWorks Search customers.

For users on LucidWorks Search for Windows Azure, the above URL would be: 'https://s-9sdf10b.azure.lucidworks.io/api/collections'.

Getting Support & Training

There are several options to get answers to questions besides this documentation:

- The [LucidWorks Search Forum](#) is a place to ask questions and share information about your implementation.
- The [LucidWorks Search KnowledgeBase](#) has articles written by our support and consulting staff around common issues and questions.
- [Training Videos](#) produced by the LucidWorks training team.
- Premium support is also available, providing access to a help desk ticketing system. For more information see [Lucene/Solr Support](#).

LucidWorks REST API Reference

In a system in which requirements remain fairly static, or at least predictable, the LucidWorks user interface may be sufficient for managing your system. For more complex or dynamic systems, LucidWorks Search provides programmatic, remote access to many aspects of configuration and operation through a REST API. All tasks that can be accomplished with the Admin UI can also be done with the REST API. Some of the more complex administration tasks are available through the API only.

About the LucidWorks Search REST APIs

Topics covered in this section:

- [About the LucidWorks Search REST APIs](#)
- [Quick Start](#)
- [List of Available APIs](#)

The LucidWorks Search REST APIs focus on administrative tasks of manipulating settings, fields, data sources, and other configuration options as well as some system monitoring functions. Programmatic access to search itself is by making GET requests to Solr directly, as documented in the section [Getting Search Results](#). The [Apache Solr Reference Guide](#) also contains a great deal of information about searching Solr.

The LucidWorks Search APIs use GET, POST, PUT and DELETE requests. The results of a REST request depend not only on the type of request, but on whether you called it on an object or group of objects. When you use REST requests with LucidWorks, you send a JSON request to the endpoint specified for the object (or group) you want to work with. The system then sends back the results as a JSON object. Currently, LucidWorks only recognizes JSON requests.

Windows users should take care when copying the examples as they assume that you are familiar with how to modify unix-based curl commands for the Windows environment.



About Server Addresses in the API Documentation

The LucidWorks Search REST API uses the [Core component](#), installed at `http://localhost:8888/` by default. Many examples in this Guide use `http://localhost:8888/` as the server location, so if you changed this location on install, be sure to change the destination of your REST requests.

Quick Start

A few pages give examples of using the APIs in configuring and managing a LucidWorks-based search application.

- [Getting Started Indexing](#) gives an overview of the calls required to set up a data source and index content.
- [Error Response Format](#) shows the format of errors if the calls go wrong.
- [Overview of REST API calls](#) provides a quick reference of the APIs and what they do.
- [Advanced Operations Using the REST API](#) contains examples of more advanced tasks, such as creating and editing fields, monitoring data sources, and configuring users.
- [Example Clients](#) includes three sample clients (in Perl, Python and C#) that were implemented using the REST API.

List of Available APIs

- [Version](#): Shows information about the LucidWorks and Solr versions being used by the system.

- **Collections**: Groups of documents that are logically separate.
 - **Collection Information**: Get information about the collection.
 - **Collection Templates**: Get information about templates that can be used when creating new collections.
 - **Collection Index Delete**: Delete the entire index or only data from a single data source.
 - **Activities**: Control schedules for resource-intensive operations such as optimization and building indexes.
 - **Status**: The status of currently running Activities.
 - **History**: Statistics for the last 50 runs of an Activity.
 - **Data Sources**: The conduits by which data enters LucidWorks indexes (see also the sections for individual data source types).
 - **Schedules**: Control when data is imported.
 - **Data Source Jobs**: Start and Stop data source jobs.
 - **Status**: Get the status of currently running data sources.
 - **History**: Statistics for the last 50 runs of a data source.
 - **Crawl Data**: Delete the crawling history for a specific data source.
 - **Batch Crawling**: Create and manage crawling when the data shouldn't be indexed until a later time.
 - **JDBC Drivers**: Load required JDBC drivers for database indexing.
 - **Field Types**: Create new field types or modify existing types.
 - **Fields**: How data from a single document is organized in LucidWorks indexes.
 - **Dynamic Fields**: Define dynamic fields for on-the-fly field creation based on specific patterns.
 - **Filtering Results**: Use Access Control Lists to filter results for Windows Shares.
 - **Search Components**: List active search components for a particular search handler.
 - **Settings**: Many different query- and index-time settings.
 - **Caches**: Configure how Solr caches documents to speed results.
 - **Click Scoring**: Integrate Click Scoring with your own search application to record which documents are most popular with users.
 - **Roles**: Configure search filters to control access to documents.
- **Alerts**: Create and modify alerts for users.
- **Users**: Create user accounts (if not using an external user management system, such as LDAP).
- **SSL Configuration**: Configure LucidWorks to work with SSL.
- **Crawler Status**: Status of communication with the Connectors component.

- ✔ When creating or updating a resource (using a POST or a PUT), you generally only need to include those entries in your map that you need to set. When LucidWorks creates a resource, entries you omit will get their default values. When you update a resource, entries you omit will retain their previous values.

JSON primitives can be used and will be returned, but LucidWorks also accepts string parse-able equivalents. For example, both 4 or "4" are valid inputs for an integer type, but 4 will be returned by the REST API.

APIs that take a list will also take a single variable (for example, a `list<string>` accepts string) and treat it as a list of size 1.

For convenience, you can append ".json" to any method.

Getting Started Indexing

The following examples use the LucidWorks REST API for some of the most common uses of the API, to control and monitor data sources and indexing. For example, suppose you wanted to create a data source that crawls a web site, such as <http://www.grantingersoll.com>; in general, if you were using `curl` to make your REST requests directly, the process would look like this:

1. First, create the data source:

```
curl -H "Content-Type: application/json" -d '{"url" :  
"http://www.grantingersoll.com", "crawl_depth": "2", "type": "web", "name": "Sample  
Site", "crawler": "lucid.aperture"}'  
http://localhost:8888/api/collections/collection1/datasources
```

Most of these keys, such as `url` and `type`, are obvious; check the full documentation for the keys involved in creating various types of [Data Sources](#).

The response is a JSON representation of the object you just created, which includes the `id` value:

```
{
  "id": 1,
  "collection": "collection1",
  "type": "web",
  "url": "http://www.grantingersoll.com/",
  "crawler": "lucid.aperture",
  "bounds": "tree",
  "category": "Web",
  "name": "Sample Site",
  "crawl_depth": 2,
  "max_bytes": 10485760,
  "include_paths": [

  ],
  "collect_links": true,
  "exclude_paths": [

  ] ,
  "mapping": {
    "multiVal": {
      "fileSize": true,
      "body": true,
      ...
    },
    "defaultField": null,
    "mappings": {
      "slide-count": "pageCount",
      "content-type": "mimeType",
      "body": "body",
      ...
    },
    "dynamicField": "attr",
    "types": {
      "filesize": "INT",
      "pagecount": "INT",
      "lastmodified": "DATE",
      "datecreated": "DATE",
      "date": "DATE"
    },
    "uniqueKey": "id" ,
    "datasourceField": "data_source"
  }
}
```

2. The next step is to tell LucidWorks to index this data source by creating a new job.

In this case, that means sending a PUT request for collection `collection1`, data source 1:

```
curl -X PUT http://localhost:8888/api/collections/collection1/datasources/1/job
```

(Note the -X PUT switch.)

In this case, there is no JSON object to pass; like many of the REST API calls, the important information is in the URL. In this case, you're passing the collection (`collection1`) and the data source number on which the job should be run (`1`). Check the full documentation for more information on stopping and starting [Data Source Jobs](#).

This request does not return anything, but it does start the index running.

3. To check the status, you can send a GET request:

```
curl http://localhost:8888/api/collections/collection1/datasources/1/status
```

Once again, you are passing the collection and data source number in the URL. The response tells you that the data source is still being indexed:

```
{
  "id": 1,
  "crawlStarted": "2011-03-17T16:34:45+0000",
  "numUnchanged": 0,
  "crawlState": "RUNNING",
  "crawlStopped": null,
  "jobId": "1",
  "numUpdated": 0,
  "numNew": 90,
  "numFailed": 0,
  "numDeleted": 0
}
```

When the job is complete, the response will look something like this:

```
{
  "id": 1,
  "crawlStarted": "2011-03-17T16:34:45+0000",
  "numUnchanged": 0,
  "crawlState": "FINISHED",
  "crawlStopped": "2011-03-17T28:26:06+0000",
  "jobId": "1",
  "numUpdated": 0,
  "numNew": 328,
  "numFailed": 2,
  "numDeleted": 0
}
```

4. You can also inspect the properties of the overall index itself:

```
curl http://<server address>/api/collections/collection1/info
```

Again, you are passing the collection name (`collection1`) in the URL. Check the full documentation for more information on working with [Collections](#).

This call gives you a response that shows all of the information about the collection itself:

```
{
  "free_disk_space": "12.2 GB",
  "index_last_modified": "2011-03-17T21:55:45+0000",
  "index_has_deletions": false,
  "data_dir":
"C:\\Users\\Nick\\LucidImagination\\LucidWorksSearch\\bin\\...\\solr\\cores\\collection1_0\\",
  "index_size": "1.3 MB",
  "index_directory": {
    "directory":
"C:\\Users\\Nick\\LucidImagination\\LucidWorksSearch\\solr\\cores\\collection1_0\\data\\in",
    "lockID": "lucene-87258 8020481afad83452b87f7517d46",
    "readChunkSize": 104857600,
    "lockFactory": {
      "lockDir":
"C:\\Users\\Nick\\LucidImagination\\LucidWorksSearch\\solr\\cores\\collection1_0\\data\\in",
      "lockPrefix": null
    }
  },
  "collection_name": "collection1",
  "index_is_optimized": false,
  "index_size_bytes": 1318288,
  "free_disk_bytes": 13072003072,
  "index_max_doc": 0,
  "index_num_docs": 0,
  "index_version": 1300398945085,
  "index_is_current": true,
  "root_dir": "C:\\",
  "instance_dir": "collection1_0",
  "total_disk_space": "222.7 GB",
  "total_disk_bytes": 239171792896
}
```

When you are satisfied that your data has been indexed, you are free to start executing searches. See the section [Getting Search Results](#) for details on how to construct queries and interpret the responses.

Error Response Format

In an ideal world, all of your programming calls will work perfectly. Unfortunately, we do not live in an ideal world, and occasionally you will need to deal with error messages. LucidWorks Search returns errors as JSON maps that provide all of the information you need to determine the problem. They are in the following format:

```
{
  "http_status_name": {string},
  "http_status_code": {integer},
  "errors": [
    {
      "message": {string},
      "key": {string}
    },
    ...
  ]
}
```

These values correspond to the following information:

http_status_name: The name of the status code.

http_status_code: The integer status code that classifies the error. These integer codes correspond to standard HTTP response codes. For example, if you reference an object that does not exist, the error code will be "404", the traditional "Not Found" response. For more information, see <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>.

errors: This object contains more detailed information on the reasons for the error status, including:

message: A human-readable error message explaining the problem.

key: The input key that the message pertains to. This may be an empty string if the error does not correspond to a submitted key.

Example

```
{
  "http_status_name": "Unprocessable Entity",
  "http_status_code": 422
  "errors": [
    {
      "message": "Unknown type: bad_type",
      "key": "type"
    },
    {
      "message": "start_time could not be parsed as a date",
      "key": "start_time"
    }
  ]
}
```

Note that more than one error may be passed in an error response.

Overview of REST API calls

This section provides an overview of the various API calls, what they do, with links to more documentation for each call.

- [Collection-Specific APIs](#)
 - [Collection APIs](#)
 - [Data Source & Crawler APIs](#)
 - [Activity APIs](#)
 - [Field-Related APIs](#)
 - [Setting APIs](#)
 - [User & Search-Related APIs](#)
- [Global APIs](#)

Collection-Specific APIs

These APIs all require the collection name to be specified in the request. They apply only to the collection named; changes to multiple collections require multiple API calls to each collection.

Collection APIs

API	Description
/api/collections	GET: Retrieve a list of collections for the entire installation with basic information about name and data location on disk. POST: Create a new collection.
/api/collections/collection	GET: Retrieve basic information for a specific collection. DELETE: Delete a collection.
/api/collections/collection/info	GET: Retrieve detailed information about the size and contents of a collection.
/api/collections/collection/info/name	GET: Retrieve a single data point about the collection such as the free disk space or the number of documents.
/api/collections/collection/index	DELETE: Delete the index for an entire collection.

Data Source & Crawler APIs

API	Description
/api/collections/collection/datasources	GET: Retrieve a list of all data sources and their configurations. POST: Create a new configuration.

API	Description
/api/collections/collection/datasources/id	GET: Retrieve details of the configuration for a specific data source. PUT: Update a specific data source. DELETE: Delete a data source.
/api/collections/collection/datasources/id/schedule	GET: Retrieve the schedule for a specific data source. PUT: Update the schedule for a data source.
/api/collections/collection/datasources/all/job	GET: Retrieve the crawling status of all data sources. PUT: Instruct all data sources to start crawling. DELETE: Instruct all data sources to stop crawling.
/api/collections/collection/datasources/id/job	GET: Retrieve the crawling status of a specific data source. PUT: Instruct a specific data source to start crawling. DELETE: Instruct a specific data source to stop crawling.
/api/collections/collection/datasources/id/status	GET: Retrieve the crawling status of a specific data source.
/api/collections/collection/datasources/id/history	GET: Retrieve the history of the last 50 crawls of a specific data source.
/api/collections/collection/datasources/id/crawldata	DELETE: Delete the crawl history for a data source.
/api/collections/collection/datasources/id/index	DELETE: Delete indexed content for a specific data source.
/api/collections/collection/datasources/id/mapping	GET: Get only the field mapping settings for a specific data source. PUT: Update the field mapping settings for a specific data source. DELETE: Remove all custom field mapping settings for a specific data source, and return it to defaults.
/api/collections/collection/datasources/id/mapping/part	GET: List the values for a specific part of the field mapping settings for a data source. DELETE: Remove custom values for a specific part of the field mapping settings for a data source (and return it to defaults).
/api/collections/collection/datasources/id/mapping/part/key	GET: List the values for a specific field mapping setting, within a <i>part</i> . DELETE: Remove custom values for a specific field mapping setting, within a <i>part</i> (and return it to defaults).
/api/collections/collection/jdbcd drivers	GET: Retrieve a list of all JDBC drivers. POST: Upload a new JDBC driver.

API	Description
/api/collections/collection/jdbcdrivers/filename	DELETE: Remove a JDBC driver.
/api/collections/collection/jdbcdrivers/classes	GET: Retrieve a list of all JDBC 4.0 compliant drivers.
/api/collections/collection/batches	GET: Retrieve a list of all batches. DELETE: Delete all batches.
/api/collections/collection/batches/crawler	GET: Retrieve a list of all batches for a specific crawler. DELETE: Delete all batches for a specific crawler.
/api/collections/collection/batches/crawler/job	GET: Retrieve a list of all jobs for a specific crawler. PUT: Define a job for a specific crawler.
/api/collection/collection/batches/crawler/job/batch_id	GET: Retrieve the status of a batch crawl job. PUT: Start a batch crawl job. DELETE: Stop a batch crawl job.

Activity APIs

API	Description
/api/collections/collection/activities	GET: Retrieve a list of activities and their schedules. POST: Create an activity and its schedule.
/api/collections/collection/activities/id	GET: Retrieve the schedule for a specific activity. PUT: Update the schedule for a specific activity. DELETE: Delete the schedule for a specific activity.
/api/collections/collection/activities/id/status	GET: Retrieve the status of a specific activity.
/api/collections/collection/activities/id/history	GET: Retrieve the history of the last 50 runs of a specific activity.

Field-Related APIs

API	Verb & Description
/api/collections/collection/fields	GET: Retrieve all fields and their attributes. POST: Create a new field.
/api/collections/collection/fields/name	GET: Retrieve attributes for a specific field. PUT: Update the attributes for a specific field. DELETE: Delete a field.

API	Verb & Description
/api/collections/collection/fieldtypes	GET: Retrieve all field types and their attributes. POST: Create a new field types.
/api/collections/collection/fieldtypes/name	GET: Retrieve attributes for a specific field type. PUT: Update the attributes for a specific field type. DELETE: Delete a field type.
/api/collections/collection/dynamicfields	GET: Retrieve all dynamic fields and their attributes. POST: Create a new dynamic field.
/api/collections/collection/dynamicfields/name	GET: Retrieve attributes for a specific dynamic field. PUT: Update the attributes for a specific dynamic field. DELETE: Delete a dynamic field.

Setting APIs

API	Verb & Description
/api/collections/collection/settings	GET: Retrieve all settings for a collection. PUT: Update settings for a collection.
/api/collections/collection/settings/name	GET: Retrieve a specific setting. PUT: Update a specific setting.
/api/collections/collection/caches	GET: Retrieve details of all configured caches. POST: Create a new cache configuration.
/api/collections/collection/caches/name	GET: Retrieve details of a specific cache. PUT: Update details of a specific cache DELETE: Delete a cache configuration.

User & Search-Related APIs

API	Verb & Description
/api/collections/collection/click	GET: Retrieve statistics about recent Click Scoring events. PUT: Record Click Scoring events.
/api/collections/collection/roles	GET: Retrieve a list of all existing roles. POST: Create a new role.

API	Verb & Description
/api/collections/collection/roles/role	GET: Retrieve details for a specific role. PUT: Update details for a specific role. DELETE: Delete a role.
/api/collections/collection/filtering	GET: Retrieve a list of existing search filter configurations. POST: Create a new search filter configuration.
/api/collections/collection/filtering/instance	GET: Retrieve details of a specific search filter configuration. PUT: Update the details of a specific search filter configuration. DELETE: Delete a specific search filter configuration.
/api/collections/collection/components/list-name?handlerName=/handlerName	GET: List the search components for a search handler. PUT: Update the search components for a search handler.

Global APIs

These APIs are not specific to a collection, but instead apply to the entire LucidWorks installation.

API	Verb & Description
/api/collectiontemplates	GET: Retrieve a list of available collection templates.
/api/version	GET: Retrieve the Solr and LucidWorks version of the application.
/api/alerts	GET: Retrieve all alerts and their attributes. POST: Create an alert.
/api/alerts?username=username	GET: Retrieve all alerts for a specific user.
/api/alerts/id	GET: Retrieve details for a specific alert. PUT: Update details for a specific alert. DELETE: Delete a specific alert.
/api/alerts/id/check	PUT: Run an alert to check for new results.
/api/users	GET: Retrieve a list of all locally-created users. POST: Create a user locally.

API	Verb & Description
/api/users/username	GET: Retrieve details for a specific user. PUT: Update details for a specific user. DELETE: Delete a user.
/api/config/ssl	GET: Retrieve details of the current SSL implementation. PUT: Update the SSL implementation.
/api/crawlers/status	GET: Get status and history of communication between the Core and Connectors components.

Pretty Printing JSON Output

JSON is easy for computers to read, but not so easy for people. If you'd like to have the JSON responses from your API calls nicely formatted when running on the command line, there are a few options.

- [Python's json.tool](#)
- [Custom Shell Script Using Python](#)
- [Browser Options](#)

Python's json.tool

If you have Python installed on the server running the LucidWorks Search [Core component](#), you can usually just pipe the output to Python's `json.tool` by adding `python -m json.tool` to the end of every API call. For example:

```
curl http://localhost:8888/api/collections/collection1/settings | python -m json.tool
```

Custom Shell Script Using Python

1. Install Python's [setuptools](#) per the instructions in the link. This will install a tool called `easy_install` which includes `simplejson`. To check if you have it, `sudo easy_install simplejson`.
2. Create a python script as below, name it "pretty-json.sh", as we did, and save it somewhere convenient:

```
#!/usr/bin/env python
#// Convert JSON data to human-readable form.
#//(Reads from stdin and writes to stdout)
import sys
import simplejson as json
print json.dumps(json.loads(sys.stdin.read()), indent=4)
sys.exit(0)
```

Browser Options

Many of the browser plugins that support REST API calls, such as Chrome's [Postman](#) or Firefox's [RESTClient](#), will format the JSON output into readable form.

Version


The version API shows the LucidWorks and Solr version and build information. This information should be supplied to support when requesting assistance, as it will help identify the version being used and will speed resolution of any problems.

- [API Entry Point](#)
- [Get Version Information](#)
 - [Input](#)
 - [Output](#)
 - [Examples](#)

API Entry Point

/api/version: [show](#) the version information

Get Version Information

 GET /api/version

Input

Path Parameters

None.

Query Parameters

None.

Output

Output Content

The output will be in two sections labeled **solr** and **lucidworks**.

The Solr section contains the following information about the version of Solr embedded with LucidWorks:

Key	Description
svn.repo	Address of the Subversion repository Solr was taken from.
build.user	The user who created the build.
git.repo	Address of the Git repository.

Key	Description
build.date	The date the build was made.
solr.svn.rev	The last revision ID of the Solr build.
git.commit	The last commit ID of the Solr build.
build.host	The machine that created the build.
build.os	The operating system that created the build.

The LucidWorks section contains the following information about the LucidWorks software:

Key	Description
environment	Describes how the build was made.
version	The LucidWorks version number.
build.user	The user who created the build.
hudson.build.number	The build number.
build.date	The date the build was made.
git.commit	The last commit ID of the LucidWorks build.
build.host	The machine that created the build.
build.os	The operating system that created the build.

Examples

Input

```
curl 'http://localhost:8888/api/version'
```

Output

```
{
  "lucidworks": {
    "build.date": "2012/03/13 12:53",
    "build.host": "dev.lcimg.com",
    "build.os": "Mac OS X",
    "build.user": "hudson",
    "environment": "production",
    "git.commit": "65e9f82178a601c89fdee4f357a3ed44c78e5743",
    "hudson.build.number": "3462",
    "version": "2.1"
  },
  "solr": {
    "build.date": "2012/03/12 10:34",
    "build.host": "beast",
    "build.os": "Linux",
    "build.user": "rmuir",
    "git.commit": "79e27b53705305866239f7206d92912a9c58c289",
    "git.repo": "git@github.com:lucidimagination/lucene-solr.git",
    "solr.svn.rev": "1296914",
    "svn.repo": "http://svn.apache.org/repos/asf/lucene/dev/trunk"
  }
}
```

Collections

Data in LucidWorks Search is organized into Collections. A collection contains data that is logically distinct from data in other collections. Collections have their own [Data Sources](#), [Settings](#), [Fields](#), and [Role Mappings](#). In other words, collections are distinct except that they happen to run in the same instance of LucidWorks Search.

For more information about collections, see also [Working with Collections](#).

- [API Entry Points](#)
- [List Collection Names](#)
- [Create Collection](#)
- [List Information for Specific Collection](#)
- [Delete Collection](#)
- [Related Topics](#)

API Entry Points

/api/collections: [list](#) collections or [create](#) a new collection

/api/collections/collection: Get [details](#) or [remove](#) a collection

List Collection Names

GET /api/collections

Input

Path Parameters

None

Query Parameters

None

Output

Output Content

Key	Type	Description
name	string	The name of the collection.
instance_dir	string	Advanced: <filepath> . Displays the directory name of the collection in \$LWS_HOME/data/solr/cores.

Examples

Get a list of all collections and their locations:

Input


```
curl http://localhost:8888/api/collections
```

Output

```
[
  {
    "name": "new_collection",
    "instance_dir": "new_collection_1"
  },
  {
    "name": "collection 1",
    "instance_dir": "collection1_0"
  },
  {
    "name": "social",
    "instance_dir": "socialdata"
  }
]
```

[Back to Top](#)

Create Collection

 POST /api/collections

Input

Path Parameters

None

Query Parameters

None

Input content

Key	Type	Required	Default	Description
name	string	Yes	null	The name of the collection.

Key	Type	Required	Default	Description
template	string	No	None	The name of the collection template to use while creating the collection. To use the default set of LucidWorks configuration files, enter default.zip . See Using Collection Templates for more information on how to create and use collection templates.
instance_dir	string	No	None	Advanced: <i><filepath></i> Allows you to optionally override the name and/or location of the Solr instance directory for the collection. By default, this will be the next available collection <i>n</i> directory in <code>\$LWS_HOME/data/solr/cores</code> . File paths that are not absolute will be relative to <code>\$LWS_HOME/data/solr/cores</code> .
num_shards	integer	See description	None	Used only when running in SolrCloud mode, this attribute specifies the number of shards to split the index to after the collection is created. This parameter is required when in SolrCloud mode. This will register the new collection and its configuration files with ZooKeeper, and share the configurations with the other shards. This number should match the number of shards running to avoid errors, but it is possible to create a collection and start more shards later.
max_shards_per_node	integer	No	null	The maximum allowed number of shards per node of the cluster. This is used to limit the number of shards on each node and to help protect against too many replicas on a single node in case a node is not live at the time of collection creation. The default for this parameter is '1'.

Key	Type	Required	Default	Description
replication_factor	integer	No	null	The number of replicas to create for each shard. This number is multiplied with <code>num_shards</code> to define how many nodes will get a copy of the collection. The total of <code>num_shards</code> times <code>replication_factor</code> should not exceed the number of live nodes, since there cannot be more than one replica of a single shard on the same node. If this parameter is not defined, the <code>num_shards</code> parameter will be divided by the number of nodes to determine the replication factor for the collection (note, in Solr, the default is '1'; this parameter in LucidWorks Search behaves slightly differently).



Because collection templates are based on an `instance_dir`, it's recommended to specify either the `template` or the `instance_dir` when creating a new collection, not both parameters.

Output

Output Content

Key	Type	Description
instance_dir	string	The name of the directory that was created under <code>\$LWS_HOME/conf/solr/cores</code> that contains the configuration files.
name	string	The name of the collection.

Examples

Create a new collection called "social" and specify that the collection should be split across two shards of a SolrCloud cluster.

Input

```
curl -H 'Content-type: application/json' -d
'{"name":"social","num_shards":2,"template":"default.zip","replication_factor":1,"max_shard_size":1000000000}'
http://localhost:8888/api/collections
```


Output

```
{
  "instance_dir": "social_1",
  "name": "social"
}
```

[Back to Top](#)

List Information for Specific Collection

GET /api/collections/collection

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Output

Output Content

Key	Type	Description
name	string	The name of the collection.
instance_dir	string	The collection directory name relative to \$LWS_HOME/solr/cores.

Examples

Get the collection information for the "social" collection:

Input


```
curl http://localhost:8888/api/collections/social
```


Output

```
{
  "name": "social",
  "instance_dir": "socialdata"
}
```

[Back to Top](#)

Delete Collection

 DELETE /api/collections/collection

 To delete the index for a collection without deleting the entire collection, see the [Collection Index Delete](#) API.

Deleting a collection will delete all associated indexes and settings, but not alerts or manually created users. Use the [Alerts API](#) or the [Users API](#) to delete those.

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

None

Output

Output Content

None

Examples

Delete the "social" collection:

Input

```
curl -X DELETE http://localhost:8888/api/collections/social
```

Output

None.

Related Topics

- [Working with Collections](#)
- [Using Collection Templates](#)
- [Using SolrCloud in LucidWorks](#)

Collection Info

The Collection Info API can be used to retrieve some statistics about a particular collection. You can use this API to retrieve all data for a collection, or just a specific key, such as the `data_dir` or `free_disk_space`.


- [API Entry Points](#)
- [Get All Information About a Collection](#)
- [Get Specific Information About a Collection](#)

API Entry Points

`/api/collections/collection/info`: [get](#) all info about the collection.

`/api/collections/collection/info/name`: [get](#) specific info about the collection.

Get All Information About a Collection

 GET `/api/collections/collection/info`

Returns all information about the collection.

Input

Path Parameters

Key	Description
collection	the collection name

Query Parameters

None

Output

Output Content

Key	Type	Description
collection_name	string	The name of collection.
data_dir	string	The directory where the index and other data resides.
free_disk_space	string	A human-readable string indicating the amount of free disk space on the partition where the index resides.
free_disk_bytes	64-bit integer	The total number of bytes available on the partition where the index resides.

Key	Type	Description
index_is_current	boolean	Is true unless the index on disk has changes not yet visible by this instance.
index_directory	string	The current Directory implementation being used for this index.
index_has_deletions	boolean	Is true if the index has deleted documents since the last optimization.
index_last_modified	date string	The date and time that the index was last modified (1-second resolution).
index_max_doc	64-bit integer	The largest doc ID in the index.
index_num_docs	64-bit integer	The number of documents in the index.
index_is_optimized	boolean	Is true if the index was optimized after the last document was indexed.
index_size	64-bit integer	A human-readable string indicating the total size of the index.
index_size_bytes	64-bit integer	The exact size of the index in bytes.
index_version	64-bit integer	A version stamp for the index.
instance_dir	string	The home directory for the collection.
root_dir	string	The root directory of the partition on which the index resides.
total_disk_space	string	A human-readable string indicating the amount of total disk space on the partition where the index resides.
total_disk_bytes	64-bit integer	The number of total bytes on the partition where the index resides.

Examples

Input


```
curl http://localhost:8888/api/collections/collection1/info
```

Output

```
{
  "free_disk_space": "10.7 GB",
  "index_last_modified": "2011-03-18T03:46:59+0000",
  "index_has_deletions": true,
  "data_dir":
"C:\\Users\\Nick\\LucidImagination\\LucidWorksEnterpriseDocs\\bin\\..\\solr\\cores\\collection1"
  "index_size": "3.8 MB",
  "index_directory": "org.apache.lucene.store.MockDirectoryWrapper",
  "collection_name": "collection1",
  "index_is_optimized": false,
  "index_size_bytes": 3990150,
  "free_disk_bytes": 11491110912,
  "index_max_doc": 169,
  "index_num_docs": 136,
  "index_version": 1300398945104,
  "index_is_current": true,
  "root_dir": "C:\\",
  "instance_dir": "collection1_0",
  "total_disk_space": "222.7 GB",
  "total_disk_bytes": 239171792896
}
```

[Back to Top](#)

Get Specific Information About a Collection

 GET /api/collections/collection/info/name

Input

Path Parameters

Key	Description
collection	the collection name
name	the name of the info key

Query Parameters

None

Input Content

When requesting collection info, you can pass a comma-separated list of keys (such as index_current,index_version) rather than separate requests for each key.

Key	Type	Description
collection_name	string	The name of collection.
data_dir	string	The directory where the index and other data resides.
free_disk_space	string	A human-readable string indicating the amount of free disk space on the partition where the index resides.
free_disk_bytes	64-bit integer	The total number of bytes available on the partition where the index resides.
index_is_current	boolean	Is true unless the index on disk has changes not yet visible by this instance.
index_directory	string	The current Directory implementation being used for this index.
index_has_deletions	boolean	Is true if the index has deleted documents since the last optimization.
index_last_modified	date string	The date and time that the index was last modified (1-second resolution).
index_max_doc	64-bit integer	The largest doc ID in the index.
index_num_docs	64-bit integer	The number of documents in the index.
index_is_optimized	boolean	Is true if the index was optimized after the last document was indexed.
index_size	64-bit integer	A human-readable string indicating the total size of the index.
index_size_bytes	64-bit integer	The exact size of the index in bytes.
index_version	64-bit integer	A version stamp for the index.
instance_dir	string	The home directory for the collection.
root_dir	string	The root directory of the partition on which the index resides.
total_disk_space	string	A human-readable string indicating the amount of total disk space on the partition where the index resides.
total_disk_bytes	64-bit integer	The number of total bytes on the partition where the index resides.

Output

Output Content

For a list of the keys, see [GET: Output Content](#).

Examples

Request the number of documents and index version for the collection.

Input

```
curl
http://localhost:8888/api/collections/collection1/info/index_version,index_num_docs
```

Output

```
{
  "index_num_docs":136,
  "index_version":1300398945104
}
```


Collection Templates

This API lists available collection templates for use in creating new collections.

Each template is a .zip file that consists of LucidWorks configuration files. The default LucidWorks configuration is available as `default.zip` found in `$LWS_HOME/app/collection_templates`. The default configuration can be customized as needed and put in a .zip archive for use when creating new collections in the future. The .zip file can have any name, including `default.zip`, although using the same name would overwrite the system default template, meaning it would not be available at a later time if needed. All templates must be placed in `$LWS_HOME/conf/collection_templates` to be available during collection creation.

For more information about creating templates, see [Using Collection Templates](#).



Information for LucidWorks Search in the Cloud Users

You can create custom templates with LucidWorks Search **on-premise only**. Customers using LucidWorks Search hosted in AWS or Azure have two collection templates out of the box that can be accessed using this API, but there is no mechanism to create and upload custom templates.

API Entry Points

`/api/collectiontemplates`: [list](#) available collection templates

List Collection Templates

GET `/api/collectiontemplates`

Input

Path Parameters

None.

Query Parameters

None.

Output

Output Content

A JSON array of available template file names.

Examples

Input

```
curl http://localhost:8888/api/collectiontemplates
```

Output

```
[ "default.zip" ]
```

Collection Index Delete

The Collections Index API is used to delete the index for an entire collection or only documents associated with a specific data source.


- [API Entry Points](#)
- [Delete the Index for a Collection](#)
- [Delete Indexed Data for a Data Source](#)

API Entry Points

`/api/collections/collection/index`: delete the collection's index

`/api/collections/collection/datasources/datasource/index`: delete indexed content for the data source

Delete the Index for a Collection

 DELETE `/api/collections/collection/index`

Stops all running data sources, clears the main search index for the collection, and deletes all crawl history for the collection. It requires a key to be entered as a parameter to protect against accidental deletes.

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

Key	Description
key	Always set to 'iaccepttherisk'.

Output

Output Content

None

Examples

Input


```
curl -X DELETE
http://localhost:8888/api/collections/collection1/index?key=iaccepttherisk
```

Output

None.

[Back to Top](#)

Delete Indexed Data for a Data Source

 DELETE /api/collections/collection/datasources/id/index

Stops the specified data source, deletes all documents from the collection's search index, and deletes all history from crawling activities. If the data source is SolrXML it must have been configured to include data source metadata before the contents can be deleted with this API. Otherwise, you will need to delete all documents in the collection to delete these documents.

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Output

Output Content

None

Examples

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/datasources/8/index
```

Output

None.

Activities

Activities are scheduled events that perform intensive operations on a collection, such as optimizing the index or creating the [auto-complete](#) index. Typically, schedules are recurring so that LucidWorks Search performs these activities periodically. However, activities can also be started at with this API for a single run. Activities controlled by this API are:

- Optimizing the index
- Indexing autocomplete data
- Processing click-scoring boost data

Schedules can be automatically disabled by LucidWorks Search if they fail on a consistent basis, but not all failures will trigger a deactivation of the schedule. If there is a missing parameter or other fatal error in the configuration of the task that will always cause an error, the schedule will be deactivated. Similarly, if there is another circumstance that causes the activity to fail when it launches (such as, the Solr handler is missing), and the scheduled task fails three times, the schedule will be deactivated. If a task consistently runs for a time and fails at some point, that will not trigger deactivation of the schedule.


- [API Entry points](#)
- [Get a List of Activities](#)
- [Create an Activity](#)
- [View a Specific Activity](#)
- [Update an Activity's Schedule](#)
- [Delete a Scheduled Activity](#)
- [Related Topics](#)

API Entry points

`/api/collections/collection/activities`: get a [list](#) of activities, or [create](#) a new one

`/api/collections/collection/activities/id`: [view](#), [delete](#) or [update](#) an existing activity

Get a List of Activities

 GET `/api/collections/collection/activities`

Note that activities that have never been run will not show in this list.

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Output

Output Content

A list of activities. For each activity, the fields are:

Key	Type	Description
start_time	date string	The start date and time for this schedule, in the format <code>yyyy-MM-dd'T'HH:mm:ss' +/- 'hhmm</code> . The '+/-' is adjusted for the time zone relative to UTC.
period	64-bit integer	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	The type of activity: optimize , click , autocomplete .
active	boolean	If true , this schedule will be run at the next scheduled time.

Activity Types

Activity	Description
optimize	Optimizes the index.
click	Processes logs of user clicks to calculate boost values.
autocomplete	Runs the auto-complete source build phase.

Examples

Get a list of all the active activities for the collection.

Input

```
curl http://localhost:8888/api/collections/collection1/activities
```


Output

```
[
  {
    "id": 9,
    "active": true,
    "start_time": "2011-03-18T04:44:39+0000",
    "type": "optimize" ,
    "period": 86400
  },
]
```

```
{
  "id": 11,
  "active": true,
  "start_time": "2011-03-1 8T04:45:03+0000",
  "type": "autocomplete",
  "period": 86400
}
```

[Back to Top](#)

Create an Activity

 POST /api/collections/collection/activities

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input Content

Key	Type	Required	Default	Description
start_time	date string	Yes	null	The start date and time for this schedule, in the format <code>yyyy-MM-dd'T'HH:mm:ss' +/- 'hhmm</code> . The API can accept a relative time of "now". The 'T' is entered without quotes. The '+/-' is for the time zone relative to UTC, and also entered without quotes. If preferred, an integer can be entered instead, which will calculate a start time based on that number of seconds from "now". So, for example, <code>"start_time":120</code> would start the activity 2 minutes (120 seconds) from when the activity was scheduled.
period	64-bit integer	No	0	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	Yes	null	The type of activity: optimize , click , autocomplete .
active	boolean	No	false	

Key	Type	Required	Default	Description
				If true , this schedule will be run at the next scheduled time.

Output

Output Content

See the section on [listing activities](#) for definitions of fields output.

Examples

Input

```
curl -H 'Content-type: application/json' -d '{"period": 0,"type":  
"optimize","start_time": "2011-03-29T12:10:32-0700","active": true}'  
http://localhost:8888/api/collections/collection1/activities
```

Output

```
{  
  "id":19,  
  "active":true,  
  "start_time":"2011-03-29T19:10:32+0000",  
  "type":"optimize",  
  "period":0  
}
```

[Back to Top](#)

View a Specific Activity

 GET /api/collections/collection/activities/id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The activity's ID.

Query Parameters

None

Input Content

None

Output

Output Content

Key	Type	Description
start_time	date string	The start date and time for this schedule, in the format <code>yyyy-MM-dd'T'HH:mm:ss' +/- 'hhmm</code> . The '+/-' is adjusted for the time zone relative to UTC.
period	64-bit integer	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	The type of activity: optimize , click , autocomplete .
active	boolean	If true , this schedule will be run at the next scheduled time.

Examples

Input


```
curl http://localhost:8888/api/collections/collection1/activities/19
```

Output

```
{
  "id":19,
  "active":true,
  "start_time":"2011-03-29T19:10:32+0000",
  "type":"optimize",
  "period":0
}
```

[Back to Top](#)

Update an Activity's Schedule

 `PUT /api/collections/collection/activities/id`

Input

Path Parameters

Key	Description
collection	The collection name

Key	Description
id	The activity's ID.

Query Parameters

None

Input Content

Key	Type	Description
start_time	date string	The start date and time for this schedule, in the format <code>YYYY-MM-dd 'T' HH:mm:ss ' +/- ' hhmm</code> . The API can accept a relative time of "now". The 'T' is entered without quotes. The '+/-' is for the time zone relative to UTC, and also entered without quotes. If preferred, an integer can be entered instead, which will calculate a start time based on that number of seconds from "now". So, for example, <code>"start_time":120</code> would start the activity 2 minutes (120 seconds) from when the activity was scheduled.
period	64-bit integer	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	The type of activity: optimize , click , autocomplete .
active	boolean	If true , this schedule will be run at the next scheduled time.

Activity Types

Activity	Description
optimize	Optimizes the index.
click	Processes logs of user clicks to calculate boost values.
autocomplete	Runs the auto-complete source build phase.

Output

Output Content

None

Examples

Set the server to optimize the index once an hour.

Input


```
curl -X PUT -H 'Content-type: application/json' -d '{"period": 6000}'  
http://localhost:8888/api/collections/collection1/activities/19
```

Output

None. (Check properties to confirm changes.)

[Back to Top](#)

Delete a Scheduled Activity

 DELETE /api/collections/name/activities/id

Input

Path Parameters

Key	Description
collection	The collection name
id	The activity's ID

Query Parameters

None

Output

Output Content

None

Examples

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/activities/19
```

Output

None. (Check properties to confirm changes)

[Back to Top](#)

Related Topics

- [Click Scoring Relevance Framework](#)
- [Auto-Complete of User Queries](#)

Activity Status


The Activities Status API allows you to get information about whether or not an Activity is currently running.

- [API Entry points](#)
- [Get the Current Status of an Activity](#)

API Entry points

`/api/collections/collection/activities/id/status`: get the status of an activity.

Get the Current Status of an Activity

 GET `/api/collection/collection/activities/id/status`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The activity ID. Use 'all' for all activities.

Query Parameters

None

Input Content

None

Output

Output Content

Key	Type	Description
id	int	The activity ID.
running	boolean	True indicates the activity is currently running.
type	string	The activity type. Possible types are click , autocomplete , or optimize .

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/activities/2/status
```

Output

While the activity is running:

```
{
  "id" : 2,
  "running" : true,
  "type" : "optimize"
}
```

Once process is finished, and the activity is idle:

```
{
  "id" : 2,
  "running" : false,
  "type" : "optimize"
}
```

Activity History


The Activity History API provides a means to get the historical statistics for previous Activity runs.

- [API Entry points](#)
- [Get the History of a Data Source](#)

API Entry points

`/api/collections/name/activities/id/history`: get statistics for the last 50 runs of the given Activity.

Get the History of a Data Source

 GET `/api/collections/collection/activities/id/history`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The activity ID. Use 'all' for all activities.

Query Parameters

None

Input Content

None

Output

Output Content

Key	Type	Description
history	JSON map	Contains the following two fields (<code>activity_started</code> and <code>activity_finished</code>) in a JSON map.
activity_started	date string	When the activity began.
activity_finished	date string	When the activity finished.
id	integer	The ID of the activity, if the API call was not for a specific activity.

Examples

Get a history of all activities:

Input

```
curl http://localhost:8888/api/collections/collection1/activities/all/history
```

Output

```
[
  {
    "history": [
      {
        "activity_finished": "2011-03-18T04:44:39+0000",
        "activity_started": "2011-03-18T04:44:39+0000"
      }
    ],
    "id": 9
  },
  {
    "history": [
      {
        "activity_finished": "2011-03-18T0 4:44:44+0000",
        "activity_started": "2011-03-18T04:44:44+0000"
      }
    ],
    "id": 10
  },
  {
    "history" : [
      {
        "activity_finished": "2011-03-18T04:45:03+0000",
        "activity_started": "2011-03-18T0 4:45:03+0000"
      }
    ],
    "id": 11
  }
]
```

Get the history for activity number 9:

Input

```
curl http://localhost:8888/api/collections/collection1/activities/9/history
```

Output

```
[  
  {  
    "activity_finished": "2011-03-18T04:44:39+0000",  
    "activity_started": "2011-03-18T04:44:39+0000"  
  }  
]
```


Data Sources

Data sources describe a target repository of documents and how to access documents in the repository. The definitions include the location of the repository, any authentication credentials that are required, and how to handle links to other documents or available paths on a filesystem tree. Instructions for how to handle the output of the job and how to handle errors can also be provided. This description is then used to create a crawl job to be executed by a crawler.

The available parameters are defined by the data source type and available types are defined by the crawler, also referred to as a *crawler controller*.

A data source is defined by selecting a crawler controller, then specifying a valid type for that crawler. Some crawlers work with several types, while others only support a single type. It is important to match the correct crawler with the type of data source to be indexed. If the type specified is not allowed by the crawler, the API will return an error.

All data sources share some attributes (parameters), but each one also has attributes specific to that type. Review the supported attributes carefully when creating data sources with the API.

For an introduction to how crawling works in LucidWorks Search, please see the section [Overview of Crawling](#).

In this section:

- [API Endpoints](#)
- [Get a List of Data Sources](#)
- [Create a Data Source](#)
- [Get Data Source Details](#)
- [Update a Data Source](#)
- [Delete a Data Source](#)

At present, the LucidWorks Search includes the following built-in crawler controllers that support the following kinds of data sources:

Crawler Controller	Symbolic Name	Data Source Types Supported
Aperture-based crawlers	lucid.aperture	<ul style="list-style-type: none">• Local file systems (LucidWorks Search on-premise only)• Websites
DataImportHandler-based JDBC crawler	lucid.jdbc	<ul style="list-style-type: none">• JDBC databases (LucidWorks Search on-premise only)

SolrXML crawler	lucid.solrxml	<ul style="list-style-type: none"> • Solr XML files (LucidWorks Search on-premise only)
Google Connector Manager-based crawler	lucid.gcm	<ul style="list-style-type: none"> • Microsoft SharePoint servers (Microsoft Office SharePoint Server 2007, Microsoft Windows SharePoint Services 3.0, SharePoint 2010)
Remote file system and pseudo-file system crawler	lucid.fs	<ul style="list-style-type: none"> • Simple Filesystem Data Sources • SMB / CIFS (Windows Shares) filesystems • Hadoop Distributed File Systems (HDFS) • Amazon S3 buckets (also known as "S3 native") • HDFS over Amazon S3 • FTP servers
Hadoop crawler	lucid.hadoop.*	<ul style="list-style-type: none"> • Apache Hadoop v1.x • Apache Hadoop v2.x • Cloudera CDH • Intel Distribution for Hadoop • MapR Hadoop • Pivotal Hadoop
MongoDB crawler	lucid.mongodb	<ul style="list-style-type: none"> • MongoDB instances
Twitter search	lucid.twitter.search	<ul style="list-style-type: none"> • Twitter Searches using Twitter's search API
Twitter stream	lucid.twitter.stream	<ul style="list-style-type: none"> • Twitter Streams using Twitter's stream API
Azure Blob crawler	lucid.azure_blob	<ul style="list-style-type: none"> • Azure Blob storage containers
Azure Table crawler	lucid.azure_table	


Crawler Controller	Symbolic Name	Data Source Types Supported
		<ul style="list-style-type: none">Azure Table instances
Push crawler	lucid.push	<ul style="list-style-type: none">Externally generated data pushed to LucidWorks and Solr

API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

Get a List of Data Sources

 GET `/api/collections/ collection /datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

A JSON map of attributes to values. The exact set of attributes for a particular data source depends on the `type`. There is, however, a set of attributes common to all data source types. Specific attributes are discussed in sections for those types, linked below.

Common Attributes

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the instance. For example, if LucidWorks has been installed in the default location and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 2, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the Zook again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to

Key	Type	Required	Default	Description
				dynamicField_sourceName, after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as data_source and data_source_type) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p>

Key	Type	Required	Default	Description
				<p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING

unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also

Key	Type	Required	Default	Description
				be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	

Key	Type	Required	Default	Description
				When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Attributes for Specific Data Source Types

Each data source type has attributes specifically for that type. To find the attributes for a specific data source type, see the API documentation for that type. The types are:

- [Website](#)
- [Local Filesystem](#)
- [Databases with JDBC](#)
- [SolrXML](#)
- [Azure Blob](#)
- [Azure Table](#)
- [Hadoop Distributed Filesystem](#)
- [Amazon S3 Bucket](#)
- [Hadoop over S3](#)
- [MapR filesystem](#)
- [MapR High Volume](#)
- [FTP Server](#)
- [MongoDB](#)
- [Local Filesystem](#) (with thread control)
- [Twitter Stream](#)
- [High-Volume HDFS](#)
- [Push](#)

Examples


Input

```
curl http://localhost:8888/api/collections/collection1/datasources
```

Output

JSON output of all configured data sources.

Create a Data Source

 POST /api/collections/ collection /datasourcesInput

Path Parameters

Key	Description
collection	The collection name

Query Parameters

None

Input content

JSON block with all attributes. The ID field should not be included because it will be automatically generated by the system. See attributes in section on [getting a list of data sources](#).Output

Output Content

JSON representation of new data source. Attributes returned are listed in the section on [getting a list of data sources](#).Examples

Create a data source that includes the content of the LucidWorks web site. To keep the size down, only crawl two levels, and do not index the blog tag links or any search links. Also, do not wander off the site and index any external links.

Input

```
curl -H 'Content-type: application/json' -d
'{"crawler":"lucid.aperture","type":"web","url":"http://www.lucidworks.com/","crawl_depth"
Website}' http://localhost:8888/api/collections/collection1/datasources
```

Output


```
{
  "add_failed_docs": false,
  "auth": [],
  "bounds": "tree",
  "caching": false,
  "category": "Web",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": 1,
  "crawler": "lucid.aperture",
  "exclude_paths": [],
  "fail_unsupported_file_types": false,
  "id": 3,
```

```
"ignore_robots": false,
"include_paths": [],
"indexing": true,
"log_extra_detail": false,
"mapping": {
  "datasource_field": "data_source",
  "default_field": null,
  "dynamic_field": "attr",
  "literals": {},
  "lucidworks_fields": true,
  "mappings": {
    "acl": "acl",
    "author": "author",
    "batch_id": "batch_id",
    "body": "body",
    "content-encoding": "characterSet",
    "content-length": "fileSize",
    "content-type": "mimeType",
    "contentcreated": "dateCreated",
    "contentlastmodified": "lastModified",
    "contributor": "author",
    "crawl_uri": "crawl_uri",
    "created": "dateCreated",
    "creator": "creator",
    "date": null,
    "description": "description",
    "filelastmodified": "lastModified",
    "filename": "fileName",
    "filesize": "fileSize",
    "fullname": "author",
    "fulltext": "body",
    "keyword": "keywords",
    "last-modified": "lastModified",
    "last-printed": null,
    "lastmodified": "lastModified",
    "lastmodifiedby": "author",
    "links": null,
    "messagesubject": "title",
    "mimetype": "mimeType",
    "name": "title",
    "page-count": "pageCount",
    "pagecount": "pageCount",
    "plaintextcontent": "body",
    "plaintextmessagecontent": "body",
    "slide-count": "pageCount",
    "slides": "pageCount",
    "subject": "subject",
    "title": "title",
    "type": null,
    "url": "url"
```

```
    },
    "multi_val": {
      "acl": true,
      "author": true,
      "body": false,
      "dateCreated": false,
      "description": false,
      "fileSize": false,
      "mimeType": false,
      "title": false
    },
    "original_content": false,
    "types": {
      "date": "DATE",
      "datecreated": "DATE",
      "filesize": "LONG",
      "lastmodified": "DATE"
    },
    "unique_key": "id",
    "verify_schema": true
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "name": "Lucid Website",
  "parsing": true,
  "proxy_host": "",
  "proxy_password": "",
  "proxy_port": -1,
  "proxy_username": "",
  "type": "web",
  "url": "http://www.lucidworks.com/",
  "verify_access": true,
  "warn_unknown_mime_types": false
}
```

Get Data Source Details

 GET /api/collections/ collection /datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name

Query Parameters

Key	Type	Description
id	string	The data source ID

Input content

NoneOutput

Output Content

Attributes returned are listed in the section on [getting a list of data sources](#). Examples

Get all of the parameters for data source 6, created in the previous step.

Input

```
curl http://localhost:8888/api/collections/collection1/datasources/6
```


Output

```
{
  "id": 6,
  "max_bytes": 10485760,
  "include_paths": [
    "http://www\\.lucidworks\\.com/.*"
  ],
  "collect_links": true,
  "exclude_paths": [
    "http://www\\.lucidworks\\.com/blog/tag/.*",
    "http://www\\.lucidworks\\.com/search\\/?.*"
  ],
  "mapping": {
    "multi_val ": {
      "fileSize": true,
      "body": true,
      "author": true,
      "title": true,
      "keywords": true,
      "subject": true,
      "description": false,
      "fileName": true,
```

```
        "dateCreated": false,
        "attr": true,
        "creator": true
    },
    "default_field": null,
    "mappings": {
        "slide-count": "pageCount",
        "content- type": "mimeType",
        "body": "body",
        "slides": "pageCount",
        "subject": "subject",
        "plainte xtmessagecontent": "body",
        "lastmodifiedby": "author",
        "content-encoding": "character Set",
        "type": null,
        "date": null,
        "creator": "creator",
        "author": "author",
        "title": "titl e",
        "mimetype": "mimeType",
        "created": "dateCreated",
        "plaintextcontent": "body",
        "page count": "pageCount",
        "contentcreated": "dateCreated",
        "description": "description",
        "contributor": "author",
        "name": "title",
        "filelastmodified": "lastModified",
        "fullname" : "author",
        "fulltext": "body",
        "messagesubject": "title",
        "last-modified": "lastModified",
        "keyword": "keywords",
        "contentlastmodified": "lastModified",
        "last-printed": null,
        "links": null,
        "batch_id": "batch_id",
        "crawl_uri": "crawl_uri",
        "filesize": "fileSize",
        "page-count": "pageCount",
        "content-length": "fileSize",
        "filename": "fileName"
    },
    " dynamic_field": "attr",
    "types": {
        "filesize": "INT",
        "pagecount": "INT",
        "lastmodified": "DATE",
        "datecreated": "DATE",
        "date": "DATE"
```

```
    },
    "unique_key": "id",
    "datasource_field": "data_source"
  },
  "collection": "collection1",
  "type": "web",
  "url": "http://www.lucidworks.com/",
  "crawler": "lucid.aperture",
  "bounds": "tree",
  "category": "Web" ,
  "name": "LucidWorks Website",
  "crawl_depth": 2
}
```

Update a Data Source

 PUT /api/collections/ collection /datasources/ idInput

Path Parameters

Key	Description
collection	The collection name
id	The data source ID

Query Parameters

None

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (ID) cannot be updated. Output

Output Content

NoneExamples

Change the web data source so that it crawls three levels instead of just two:


Input

```
curl -X PUT -H 'Content-type: application/json' -d '{"crawl_depth": 3}'
http://localhost:8888/api/collections/collection1/datasources/6
```


Output

None. Check data source properties to confirm changes.

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/ collection /datasources/ idInput

Path Parameters

Key	Description
collection	the collection name
id	The data source ID

Query Parameters

None

Input content

NoneOutput

Output Content

NoneExamples

Input

```
curl -X DELETE -H 'Content-type: application/json'
http://localhost:8888/api/collections/collection1/datasources/13
```

Output

None. Check the listing of data sources to confirm deletion.

Amazon S3 Data Sources

The Amazon S3 data source type allows crawling documents stored in an Amazon S3 bucket.

As of v2.7, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

- [Amazon S3 Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [S3 Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Amazon S3 Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an S3 data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are: <ul style="list-style-type: none">• file for a filesystem (remote or local, but must be paired with the correct crawler, as below)• web for HTTP or HTTPS web sites• jdbc for a JDBC database

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the conter

Key	Type	Required	Default	Description
				<p>to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>.</p> <ul style="list-style-type: none"> • com.lucid.sda.hbase.lws.HBaseUpdateCon: The output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output locations are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the default location and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput is increased when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a comma with no whitespace between them. For example: <code>"output_args": "buffer=2,threads=10"</code> (and will use the default Solr location). If a value is missing, the default is used. • <code>output_type</code> is "com.lucid.crawl.impl.FileUpdate": The <code>output_args</code> must be a URI string for a file system point to either a directory (which must exist) or

Key	Type	Required	Default	Description
				<p>will be created during the crawl (which must not be the crawl). The path will be interpreted as entire paths should be used whenever possible. Relative paths should be interpreted relative to the working directory of the component, which is \$LWS_HOME.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.script.ScriptPreprocessor" The output_args are a script name, in the form "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data_source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector" output_args must be the host:port of the ZooKeeper again used with LucidWorks Big Data only. A value is interpreted by HBase, and will be similar to local <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which

Key	Type	Required	Default	Description
				means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such

Key	Type	Required	Default	Description
				as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the

Key	Type	Required	Default	Description
				<p>mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field.</p> <ol style="list-style-type: none"> 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div>Field mapping normalization is a step applied after all target names for field values</div>

Key	Type	Required	Default	Description
				<p>have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if multiValued=false and multiple values are encountered, only the

Key	Type	Required	Default	Description
				<div>first value is retained and all other values are discarded.</div>
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <div>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</div>

Key	Type	Required	Default	Description
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code> Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code> Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code> Any class that ends with <code>*LongField</code> becomes <code>LONG</code> Anything else not listed above becomes <code>STRING</code>
unique_key	string	No	"id"	<p>Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for</p>

Key	Type	Required	Default	Description
				checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

S3 Type-Specific Attributes

When creating a data source of type **s3**, the value **lucid.fs** must be supplied for the `crawler` attribute, described in the section on [common attributes](#). In releases prior to v2.1, this data source type was called "S3N".

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	If true , documents that failed to invalid formats, parsing err etc.) will be added to the index metadata that could be retrieved.

Key	Type	Required	Default	Description
				document (if it was only partially crawled) and the reason for the "parse" or "fetch" field, which points to the document.
bounds	string	No	None	Either tree to limit the crawl to a sub-tree, or "none" for no limit. If crawling <code>s3://10.0.0.50/docs/</code> option is the equivalent of <code>s3://10.0.0.50/docs/*</code> and not crawl <code>s3://10.0.0.50/other/</code> require more advanced crawl should choose none and using excludes options.
crawl_depth	32-bit integer	No	-1	How many path levels to descend. -1 indicate unlimited depth which is the default if left blank.
crawl_item_timeout	integer	No	600000	Defines the timeout accessing a single file during a crawl. If this timeout, then the crawl job to prevent the thread hanging. The default is 600000 milliseconds.
exclude_paths	list of strings	No	Empty	Regular expression patterns that will not match. If not empty then if any pattern matches its URL
include_extensions	list of strings	No	Empty	If you would like to index only file extensions that you define. This parameter will only look at the extension (such as "pdf" or "x") to recognize the file's mime type
include_paths	list of strings	No	Empty	Regular expression patterns to match URLs of files. If not empty the pattern must match to include the file. If blank, all sub-directory paths (limited by the crawl_depth).
index_directories	boolean	No	False	

Key	Type	Required	Default	Description
				By default, directories will not contain documents. If you would prefer that filesystem directories appear in search results as documents, select this option.
max_bytes	long	No	10,485,760	Defines the maximum size of documents that can be crawled. Optional, default is -1, which means no limit per document.
max_docs	integer	No	-1	Defines the maximum number of documents to crawl. If the value is -1, the crawler will crawl all found documents (in conjunction with other data source attributes which may limit crawling).
max_threads	integer	No	1	Defines the maximum number of threads the crawler should use. Optional, default is 1, which is single-threaded.
maximum_connections	integer	No	1000	Limits the number of tasks (connections) that will be created. Each document crawl is a task, and each task creates a connection to the Windows Share. If there is a large number of documents, a large number of connections may cause errors or degraded performance.
password	string	Yes	Null	Your Amazon S3 SecretAccessKey.
persisted_URIs_transaction_size	integer	No	1000	Limits the size of transaction to support indexing of persisted URIs to support incremental crawling.
type	string	True	Null	One of supported data source types. Must be consistent with the root URL's data source type, e.g., Amazon S3 data source type, etc.
url	string	Yes	Null	For S3 (Amazon S3 native), the fully-qualified URL that starts with the protocol, the name of the bucket inside the bucket. All URLs that point to directories that contain files to be crawled must end with a trailing slash. New requirement for v2.1. Both <code>AccessKeyId</code> and <code>SecretAccessKey</code> are needed. <code>AccessKeyId</code> as the username and <code>SecretAccessKey</code> as the password.

Key	Type	Required	Default	Description
				SecretAccessKey as the pass also pass these credentials as in the following format: s3n: //<username>@<password>. However, Amazon S3 credentials contain characters that are not URLs. In that case, you must escape credentials by setting the use_escaped_password properties explicitly
username	string	Yes	Null	Your Amazon S3 AccessKeyId.
verify_access	boolean	No	True	By default, LucidWorks Search verifies the data source is accessible at crawl time. To be able to create a data source without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.



The `include_paths` and `exclude_paths` attributes for all Remote File System data sources use [Java Regular Expressions](#).

Example S3 data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "tree",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "exclude_paths": [],
  "id": "712ccdc318a248cbaa03a27ee9181bbf",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
  "indexing": true,
  "mapping": {
```

```
...
},
"max_bytes": 10485760,
"max_docs": -1,
"max_threads": 1,
"maximum_connections": 1000,
"name": "Amazon S3",
"output_args": "threads=2,buffer=1",
"output_type": "solr",
"parsing": true,
"password": "AWS-SecretToken",
"persisted_URIs_transaction_size": 1000,
"type": "s3",
"url": "s3://s3crawler/test1/",
"username": "AWS-AccessKey",
"verify_access": true
}
```

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content


None

Apache Hadoop v1.x Data Sources

The Apache Hadoop v1.x data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.

 Before using the Apache Hadoop v1.x Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [Apache Hadoop v1.x Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Apache Hadoop v1.x Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Apache Hadoop v1.x Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Apache Hadoop v1.x data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the directory and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the Zook again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if <code>multiValued=false</code> and multiple values are encountered, only the first value is retained and all other values are discarded.
<code>original_content</code>	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not

Key	Type	Required	Default	Description
				created and documents are not preserved unless as a result of setting other options above.

Apache Hadoop v1.x Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.apache1** must be supplied for the **crawler** attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for \$HADOOP_HOME/bin/hadoop. If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to \$HADOOP_HOME/bin/hadoop, then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example Apache Hadoop 1.x data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.apache1",
  "hadoop_home": "/hadoop",
  "id": "a772505e00204592be0183632abfebc3",
  "job_jar": "hadoop-lws-job.jar",
  "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true"
```

```
-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
  "name": "Apache v1.x",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "type": "hadoop",
  "verify_access": false
}
```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Req
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/*</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud

Argument	Required	Description
		mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this mapperClass, the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> , where the key is any <code>Writable</code> and the value is	Sequen

Mapper Class Name	Description	Input
	text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls  
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of  
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk  
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Apache Hadoop v2.x Data Sources

The Apache Hadoop v2.x data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.



Before using the Apache Hadoop v2.x Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [Apache Hadoop v2.x Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Apache Hadoop v2.x Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Apache Hadoop v2.x Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Apache Hadoop v2.x data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the path and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must not the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lo script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not

Key	Type	Required	Default	Description
				created and documents are not preserved unless as a result of setting other options above.

Apache Hadoop v2.x Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.apache2** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for \$HADOOP_HOME/bin/hadoop. If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to \$HADOOP_HOME/bin/hadoop, then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example Apache Hadoop 2.x data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.apache1",
  "hadoop_home": "/hadoop",
  "id": "a772505e00204592be0183632abfebc3",
  "job_jar": "hadoop-lws-job.jar",
  "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true"
```



```
-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
  "name": "Apache v2.x",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "type": "hadoop",
  "verify_access": false
}
```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Req
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud

Argument	Required	Description
		mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this mapperClass, the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> , where the key is any <code>Writable</code> and the value is	Sequen

Mapper Class Name	Description	Input
	text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls  
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of  
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk  
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Azure Blob Data Sources

The Azure Blob crawler allows indexing of an Azure Blob instance. Windows Azure Blob storage allows storing large amounts of unstructured data in the Azure service.

The Azure Blob data source connects to a defined Azure Blob storage container with HTTPS and indexes the content found there. It automatically validates the container information provided; unlike other crawlers, it does not provide an option for disabling this validation. Once it has identified the blob blocks, the crawler will read them sequentially and prepare them for indexing. In the case of `image/` blobs, only metadata about the image files will be retrieved for indexing.

- [Azure Blob Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Azure Blob Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Azure Blob Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a file data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are:

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. It is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the default location and creating the data source for collection1, the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a space with no whitespace between them. For example, <code>threads=2 buffer=1</code>.

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdateCorruptor": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as entered paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorScriptProcessor": The output_args are a script name, in the form "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Azure Blob Type-Specific Attributes

When creating a data source of type **azure_blob**, the value **lucid.azureblob** must be supplied for the **crawler** attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
storage_container	string	No	null	The name of the Azure container where the Blobs are found
storage_account	string	No	null	The Azure storage account name.
token_secret	string	No	null	The Azure storage account token for authentication.
max_bytes	integer	No	10,485,760	Defines the maximum size of a document to crawl. Optional, default is 10,485,760, which is 10Mb per document.

Example Azure Blob data source (without mapping attributes):

```
{
  "caching": false,
  "category": "azureblob",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawler": "lucid.azureblob",
  "id": "4f3a5fd9a31342dc862327c77ed3e246",
  "indexing": true,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "name": "Azure Blob",
  "output_args": null,
  "output_type": "solr",
  "parsing": true,
  "storage_account": "AzureAccount",
  "storage_container": "AzureContainer",
  "token_secret": "AzureToken",
  "type": "azure_blob"
}
```


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT `/api/collections/collection/datasources/ id`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Azure Table Data Sources

The Azure Table crawler allows indexing of an Azure Table instance. Windows Azure Table storage allows storing large amounts of structured data in the Azure service.

It automatically validates the table information provided on data source creation; unlike other crawlers, it does not provide an option for disabling this validation. The crawler also does not support incremental crawling. All documents are retrieved each time the crawler runs.

- [Azure Table Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Azure Table Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Azure Table Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a file data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are: <ul style="list-style-type: none">• file for a filesystem (remote or local, but must be paired with the correct crawler, as below)• web for HTTP or HTTPS web sites

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file.

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content before it is sent to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: The crawler output will be sent to an HBase implementation in <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the instance. For example, if LucidWorks has been installed in <code>/opt/lucidworks</code> and creating the data source for collection1, the default URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput is increased when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in conjunction, express them as key=value pairs, separated by a comma with no whitespace between them. For example: <code>"output_args": "buffer=2,threads=10"</code> (and will use the default Solr location). If the location is missing, the default is used.

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist at the time of the crawl). The path will be interpreted as an absolute path. Relative paths should be used whenever possible. Relative paths are interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data_source_scripts. output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon": The output_args must be the host:port of the ZooKeeper. This is again used with LucidWorks Big Data only. A value is interpreted by HBase, and will be similar to localhost:2181. <p>If using a custom implementation of UpdateConnector, the update_connector attribute can be however you defined its use in the configuration.</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute mapping contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are

Key	Type	Required	Default	Description
				not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	

Key	Type	Required	Default	Description
				<p>If true, the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false. However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.</p>
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it

Key	Type	Required	Default	Description
				<p>exists, it will be mapped to the target field.</p> <ol style="list-style-type: none"> If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p>

Key	Type	Required	Default	Description
				<p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if multiValued=false and multiple values are

Key	Type	Required	Default	Description
				encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the</p>

Key	Type	Required	Default	Description
				Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code> Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code> Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code> Any class that ends with <code>*LongField</code> becomes <code>LONG</code> Anything else not listed above becomes <code>STRING</code>
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the

Key	Type	Required	Default	Description
				<code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming

Key	Type	Required	Default	Description
				documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Azure Table Type-Specific Attributes

When creating a data source of type **azure_table**, the value **lucid.azuretable** must be supplied for the **crawler** attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
storage_account	string	Yes	null	The Azure storage account name.
token_secret	string	Yes	null	The Azure storage account token for authentication.
tables	string	Yes	null	The Azure table to index.
table_filter_statement	string	No	null	A filter to apply to the crawl.

Example Azure Table data source (without mapping attributes):

```
{
  "caching": false,
  "category": "azuretable",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawler": "lucid.azuretable",
  "id": "d7c18c45057147a0bdf937e9af90e044",
  "indexing": true,
  "mapping": {
    ...
  },
  "name": "Azure Table",
  "output_args": null,
  "output_type": "solr",
  "parsing": true,
  "storage_account": "AzureAccount",
  "table_filter_statement": "",
  "tables": "WADLogsTable",
  "token_secret": "AzureToken",
  "type": "azure_table"
}
```


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Cloudera CDH Data Sources

The Cloudera CDH data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements. LucidWorks Search has been tested with Cloudera CDH v4.5.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.



Before using the Cloudera CDH Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [Cloudera CDH Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Cloudera CDH Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Cloudera CDH Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Cloudera CDH data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: The output will be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content before sending to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: The output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the instance. For example, if LucidWorks has been installed in <code>/opt/lucidworks</code> and creating the data source for collection1, the default URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must not the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lo script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's <code>multiValued</code> attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not

Key	Type	Required	Default	Description
				created and documents are not preserved unless as a result of setting other options above.

Cloudera CDH Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.cloudera** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for <code>\$HADOOP_HOME/bin/hadoop</code> . If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to <code>\$HADOOP_HOME/bin/hadoop</code> , then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example Cloudera CDH data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.cloudera",
  "hadoop_home": "/hadoop",
  "id": "0ad83e0772e0497f8efbbed1215e6f48",
  "job_jar": "hadoop-lws-job.jar",
  "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true"
```

```
-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
  "name": "Cloudera 3.5",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "type": "hadoop",
  "verify_access": false
}
```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Required
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/*</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud

Argument	Required	Description
		mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this mapperClass, the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> , where the key is any <code>Writable</code> and the value is	Sequen

Mapper Class Name	Description	Input
	text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

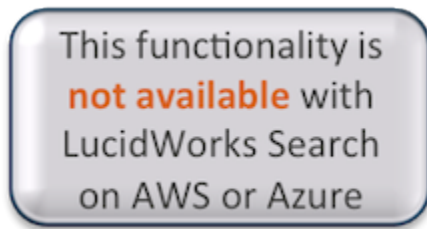
None

Output

Output Content

None

Database Data Sources



Databases can be indexed with the Database data source. The database must be accessible to the LucidWorks Search server, and they must have a valid JDBC driver available to the crawler that is appropriate for your RDBMS. Drivers are not shipped with LucidWorks, but can be loaded with either the [JDBC Drivers API](#) or the Admin UI. JDBC database data sources are available in **LucidWorks Search on-premise only**.

- [JDBC Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [JDBC Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

JDBC Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a JDBC data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are:

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the conter to Solr for indexing. Only Javascript is supporte The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateCon output will be sent to an HBase implementation <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualifie custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be se are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are p defined, <code>output_args</code> will default to the Solr ins in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in th and creating the data source for collection1, th http://127.0.0.1:8888/solr/collection1). Two a parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurr use for sending updates. This does not de use while crawling a data source, but thr updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documen before sending to SolrJ in bulk, which car reduce the number of calls to Solr. The d which means no additional buffering. In c this value to higher than one has little im performance when the number of threads 1; the performance benefits are usually s <code>threads=1</code>, but at the cost of increased J consumption. • When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdateCorruptor": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorScriptProcessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code> Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code> Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code> Any class that ends with <code>*LongField</code> becomes <code>LONG</code> Anything else not listed above becomes <code>STRING</code>
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

JDBC Type-Specific Attributes

When creating a data source of type **jdbc**, the value **lucid.jdbc** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
<code>clean_in_full_import_mode</code>	boolean	No	True	If true , the default, the equivalent of the "clean" parameter using Solr's DataImportHandler is passed, which removes documents from the index for this data source before indexing them again as new.
<code>delta_sql_query</code>	string	No	Null	This allows incremental indexing, which will only retrieve updated records in subsequent crawls without having to retrieve the entire contents of an unconstrained query. Delta queries select only primary key values of your data and must include <code>last_modified</code> condition in the following form, <code>"SELECT id FROM table WHERE last_modified > \$"</code> . The "\$" sign will hold the last successful import time from the database. If you do not use the "\$" character, the query will fail.
<code>driver</code>	string	Yes	Null	The class name of the JDBC driver.
<code>max_docs</code>	integer	No	-1	The maximum number of documents to crawl. If set to -1, all document found will be collected (in conjunction with other data source attributes which may limit the crawl).
<code>nested_queries</code>	list of strings	No	Null	If you want to index one-to-many or many-to-many relations in addition to plain rows then you can specify an arbitrary number of additional SQL "nested" queries. For example, if you want to index the list of assigned tags for your documents you can specify a query in the following form <code>"SELECT tag FROM tag INNER JOIN document_tag ON</code>

Key	Type	Required	Default	Description
				<code>document_tag.tag_id=tag.id</code> <code>WHERE document_tag.doc_id=\$".</code> The \$ sign will hold the primary key of your main data row. This query will be executed for every record of your data and corresponding list of tags will be retrieved from database and indexed in the Solr index.
password	string	Yes	Null	The password of the account that should be used to access the data.
primary_key	string	No	Null	The column name of the primary key.
sql_select_statement	string	Yes	Null	The select statement to use to generate data.
username	string	Yes	Null	The username of a database account that should be used to access the data.
url	string	Yes	Null	A URI to the database. This should be in the form of <code>jdbc:mysql://localhost/test</code> including any optional driver specific connection parameters. Database data sources are expected to be unique by URI, which means that creating two data sources for the same URI is not possible.
verify_access	boolean	No	True	By default, LucidWorks Search will attempt to verify the data source is accessible at create time. To be able to create a data source without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.

Example JDBC data source (without mapping attributes):

```
{
  "category": "Jdbc",
  "clean_in_full_import_mode": true,
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawler": "lucid.jdbc",
  "delta_sql_query": "",
  "driver": "com.mysql.jdbc.Driver",
  "id": 4,
  "mapping": {
    ...
  },
  "max_docs": -1,
  "name": "stackoverflow db",
  "nested_queries": [],
  "output_args": "http://127.0.0.1:8888/solr/collection1",
  "output_type": "solr",
  "password": "password",
  "primary_key": "id",
  "sql_select_statement": "select id, comment_text as body from comment",
  "type": "jdbc",
  "url": "jdbc:mysql://127.0.0.1/stackoverflow",
  "username": "username",
  "verify_access": true
}
```

Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source



The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

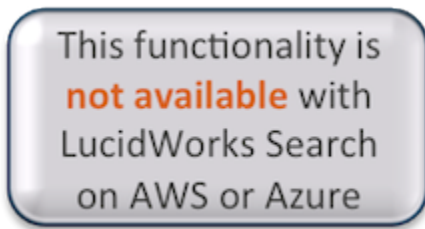
None

Output

Output Content

None

Filesystem Data Sources (Aperture)



A local file system is one that is either on the same server as the LucidWorks installation or mounted to it. Other data source types exist for remote file systems, such as [S3](#) or [Windows Shares](#). Local file system data sources are available in **LucidWorks Search on-premise only**.

LucidWorks Search includes two data sources that can be used to crawl a local filesystem. This data source type differs from the other local filesystem data source type (referred to as a [Simple Filesystem data source](#)) in that this data source uses only one thread while crawling and all content is automatically parsed by the Aperture parser (no matter what the `parsing` attribute is set to). If processing content in [batches](#), and you do not want the content parsed first, use the other filesystem data source type.

- [File Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [File Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

File Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a file data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the instance. For example, if LucidWorks has been installed in the default location and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease the number of updates while crawling a data source, but the number of updates sent to Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 2, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must not the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lo script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to

Key	Type	Required	Default	Description
				dynamicField_sourceName, after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as data_source and data_source_type) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p>

Key	Type	Required	Default	Description
				<p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING

unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also

Key	Type	Required	Default	Description
				be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	

Key	Type	Required	Default	Description
				When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

File Type-Specific Attributes

When creating a data source of type **file**, and intending a local filesystem, the value **lucid.aperture** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	<p>If true, documents that failed processing (due to invalid formats, parsing errors, IO failures, etc.) will be added to the index with whatever metadata that could be retrieved from the document (if it was only partially parsed, for example) and the reason for the error in the "parse" or "fetch" field, which would be added to the document. The default for this attribute is false.</p> <p>If this option is enabled after an initial crawl has been run, it is possible that failed documents will not be added. The reason this happens is because this crawler stores documents it has seen before (also known as persistent crawl data or crawl history) and generally skips documents seen on previous crawls if they are unchanged. To force the crawler to try to process a document again, either clear the crawl</p>

Key	Type	Required	Default	Description
				history (which will cause all documents to be considered "new" again) or update the document so it appears to the crawler to have been changed.
bounds	string	No	None	Either tree to limit the crawl to a strict subtree, or none for no limits. If tree is chosen, the crawler will only access pages using the URL as the base path. For example, if crawling <code>/Path/to/Files</code> , the tree option is the equivalent of <code>/Path/to/Files/*</code> . When <code>follow_links</code> is true, choosing none will allow the crawl to go to directories outside the base directory path entered.
crawl_depth	32-bit integer	No	-1	How many path levels to descend. Use '-1' to indicate unlimited depth, which is also the default if left empty.
exclude_paths	list of strings	No	Empty	Regular expression patterns that URLs must not match. If not empty then a file is excluded if any pattern matches its URL. This can be used to exclude certain types of files from a crawl.
fail_unsupported_file_types	boolean	No	False	If true , documents that cannot be parsed (either because of unspecified errors or because of an unknown file format) will produce an error in the logs. The default behavior is to not report these documents as failures to the log. If <code>add_failed_docs</code> is also checked, the result of these failures will also be added to the index with whatever metadata could be extracted from the

Key	Type	Required	Default	Description
				content. This is different from <code>warn_unknown_mime_types</code> in the sense that this parameter only applies to content that fails parsing and not content that doesn't have a defined mime type.
<code>follow_links</code>	boolean	No	False	Use true to instruct the crawler to follow symbolic links in the file system.
<code>include_paths</code>	list of strings	No	Empty	Regular expression patterns to match on full URLs of files. If not empty then at least one pattern must match to include a file. This could be used to limit the crawl to only certain types of files or certain subdirectories.
<code>max_bytes</code>	long	No	10485760	Defines the maximum size of any crawled file. The default is -1, which is 10Mb per document.
<code>path</code>	string	Yes	Null	The path of the directory to start reading from. Paths should be entered as the complete directory path or they will be interpreted as relative to <code>\$LWS_HOME</code> . On Unix systems, this means the path entered should start at root / level; on Windows, the drive letter and full path should be used (such as <code>C:\path</code>). Various types of relative paths, such as <code>../</code> or <code>~/</code> , are not supported. Filesystem data sources are expected to be unique by URL, which means that creating two data sources for the same directory is not possible. The API will attempt to validate that

Key	Type	Required	Default	Description
				LucidWorks Search can access the path, and will return an error if it cannot.
url	string	No	Null	Read-only value that shows the absolute path. In many cases this will be identical to the path as entered, but if the path was a shortcut or symbolic link, the url will show the real path.
verify_access	boolean	No	True	By default, LucidWorks Search will attempt to verify the data source is accessible at create time. To be able to create a data source without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.
warn_unknown_mime_types	boolean	No	False	If true , documents with no mime type specified in the format produce a warning in the log. If the file cannot be processed as plain text, it will be skipped and a warning message will be printed to the log. The default behavior is to skip these documents and not report warnings in the log. This parameter differs from <code>fail_unsupported_file_types</code> in the sense that it only applies to content that does not have a defined mime type.



The `include_paths` and `exclude_paths` attributes for File System data sources use [Java Regular Expressions](#).

Example file system data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "none",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawler": "lucid.aperture",
  "exclude_paths": [],
  "fail_unsupported_file_types": false,
  "follow_links": false,
  "id": "9e3d6defb8d1438590c2aea02237db12",
  "include_paths": [],
  "indexing": true,
  "log_extra_detail": false,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "name": "Local hard drive",
  "output_args": null,
  "output_type": "solr",
  "parsing": true,
  "path": "/Users/cassandra4work/Documents",
  "type": "file",
  "url": "file:/Users/cassandra4work/Documents/",
  "verify_access": true,
  "warn_unknown_mime_types": false
}
```


Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content


None

Delete a Data Source



The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

FTP Data Sources

Only simple FTP is supported at this time. The FTP crawler does not support crawling FTP servers that require FTPS, SFTP or FTP over SSH. By default, the FTP crawler uses passive mode; there is no setting to change to active mode at this time.

As of v2.7, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

Symbolically linked directories or files will be reported as failed documents because the FTP crawler does not follow symbolic links.

- [FTP Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [FTP Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

FTP Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an FTP data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are:

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the conter to Solr for indexing. Only Javascript is supporte The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateCon output will be sent to an HBase implementation <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualifie custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be se are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are p defined, <code>output_args</code> will default to the Solr ins in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in th and creating the data source for collection1, th http://127.0.0.1:8888/solr/collection1). Two a parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurr use for sending updates. This does not de use while crawling a data source, but thr updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documen before sending to SolrJ in bulk, which car reduce the number of calls to Solr. The d which means no additional buffering. In c this value to higher than one has little im performance when the number of threads 1; the performance benefits are usually s <code>threads=1</code>, but at the cost of increased J consumption. • When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdateCorruptor": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorScriptProcessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as data_source and data_source_type) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code> Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code> Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code> Any class that ends with <code>*LongField</code> becomes <code>LONG</code> Anything else not listed above becomes <code>STRING</code>
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

FTP Type-Specific Attributes

When creating a data source of type **ftp**, the value **lucid.fs** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	If true , documents that failed processing (due to invalid forr parsing errors, IO failures, etc) will be added to the index with whatever metadata that could be retrieved from the document (was only partially parsed, for example) and the reason for the error in the "parse" or "fetch" which would be added to the document.
bounds	string	No	None	Either true to limit the crawl to a strict subtree, or "none" for no limits. For example, if crawling <code>ftp://10.0.0.50/docs</code> , the <code>bounds</code> option is the equivalent of <code>ftp://10.0.0.50/docs/*</code> and <code>lucid.fs</code> would not crawl <code>ftp://10.0.0.50/other</code> . If you require more advanced crawl limiting, you should choose no and use the <code>include_paths</code> or <code>exclude_paths</code> options.
crawl_depth	32-bit integer	No	-1	How many path levels to descend. Use '-1' to indicate unlimited, which is also the default if left blank.
crawl_item_timeout	integer	No	600000	Defines the timeout accessing/parsing for a single file during crawl. If a file exceeds this timeout, then the crawl job will be stopped to prevent the thread from hanging for no reason. The default is 600000 milliseconds, or 10 minutes.
exclude_paths		No	Empty	

Key	Type	Required	Default	Description
	list of strings			Regular expression patterns to match on full URLs of files. If not empty, then a file is excluded if any pattern matches its URL.
include_extensions	list of strings	No	Empty	If you would like to index only files of specific file extensions that you define, list them here. This parameter will only look at the file extension (such as "pdf" or "xls") and will not recognize the file's mime types.
include_paths	list of strings	No	Empty	Regular expression patterns to match on full URLs of files. If not empty, then at least one pattern must match to include a file. If blank, all subdirectory paths will be followed (limited by the crawl_depth). This feature can be used to limit a filesystem crawl to specific subdirectories of a base directory path. For example, if the base directory path is /Path/to/Files, includes could be used to limit the crawl to specific subdirectories /Path/to/Files/Archive/2010 and /Path/to/Files/Archive/2011.
index_directories	boolean	No	False	By default, directories will not be indexed as documents. If you would prefer to have filesystem directories appear in your index as documents, select this option.
max_bytes	long	No	10,485,760	Defines the maximum size of a document to crawl. Optional, default is -1, which is 10Mb per document.
max_threads	integer	No	1	Defines the maximum number of threads the crawler should use.

Key	Type	Required	Default	Description
				Optional, default is 1, which is single-threaded. The FTP data source type does not support max_threads value greater than 1.
maximum_connections	integer	No	1000	Limits the number of tasks (connections) that will be created. Each document to be crawled creates a task, and each task creates a connection to the Windows Share system. With a large number of documents, this could create a large number of connections which may cause errors or degraded performance.
password	string	No	Null	A password to access the FTP filesystem.
persisted_URIs_transaction_size	integer	No	1000	Limits the size of transaction to store the list of persisted URIs to support incremental crawling.
url	string	Yes	Null	For FTP access, the root URL includes the protocol, ftp, the address, and the path to crawl: ftp://_host_/path/. By default, all files in the directory tree or folder hierarchy linked from the URL will be crawled, unless you select some of the limiting options available.
username	string	No	Null	A username to access the FTP filesystem. If this is left empty "anonymous" will be substituted as a name to indicate anonymous access to the FTP server.
verify_access	boolean	No	True	By default, LucidWorks Search attempts to verify the data source is accessible at create time. To be able to create a data source without verifying access, change this value to false . Note, how

Key	Type	Required	Default	Description
				that if LucidWorks Search can access the data source, it will be able to crawl it.

- ✔ The `include_paths` and `exclude_paths` attributes for all Remote File System data sources use [Java Regular Expressions](#).

Example FTP data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "tree",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "exclude_paths": [],
  "id": "75e7422d58d44fdcbfcba690342fed74",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
  "indexing": true,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "max_threads": 1,
  "maximum_connections": 1000,
  "name": "FTP Server",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "parsing": true,
  "password": "crawl@example.com",
  "persisted_URIs_transaction_size": 1000,
  "type": "ftp",
  "url": "ftp://files/test1/",
  "username": "anonymous",
  "verify_access": true
}
```

Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the `id`. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Hadoop Over S3 Data Sources

This data source is similar to the Amazon S3 data source, but has some specific handlers for Hadoop in particular. It may be possible to configure an S3 data source to crawl a Hadoop File System over S3, but this S3H data source type is the preferred method.

As of v2.7, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

This data source differs from the [High Volume HDFS](#) data source in that this data source type is designed to be polite and crawl through the content stored in Hadoop over S3 just as if it crawled a web site or any other file system. The High Volume HDFS, however, designed to take full advantage of the scaling abilities of Hadoop's Map-Reduce architecture.

- [Hadoop Over S3 \(S3H\) Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Hadoop Over S3 \(S3H\) Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Hadoop Over S3 (S3H) Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an S3H data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.

Key	Type	Required	Default	Description
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the conter to Solr for indexing. Only Javascript is supporte The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateCon output will be sent to an HBase implementation <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualifie custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be se are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are p defined, <code>output_args</code> will default to the Solr ins in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in th and creating the data source for collection1, th http://127.0.0.1:8888/solr/collection1). Two a parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurr use for sending updates. This does not de use while crawling a data source, but thr updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documen before sending to SolrJ in bulk, which car reduce the number of calls to Solr. The d which means no additional buffering. In c this value to higher than one has little im performance when the number of threads 1; the performance benefits are usually s <code>threads=1</code>, but at the cost of increased J consumption. • When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdateCorruptor": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorScriptProcessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Hadoop Over S3 (S3H) Type-Specific Attributes

When creating a data source of type **s3h**, the value **lucid.fs** must be supplied for the **crawler** attribute, described in the section on [common attributes](#). In releases prior to v2.1, this data source type was called "S3".

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	If true , documents that failed (due to invalid formats, parsing failures, etc.) will be added to whatever metadata that could be extracted from the document (if it was successfully parsed, for example) and the error in the "parse" or "fetch" operation would be added to the document.
bounds	string	No	None	Either tree to limit the crawl to a subtree, or "none" for no limit. If crawling <code>s3://10.0.0.50/docs</code> and the option is the equivalent of <code>s3://10.0.0.50/docs*</code> and then it will not crawl <code>s3://10.0.0.50/other</code> . More advanced crawling options require more advanced crawling options. You should choose none and using excludes options.
crawl_depth	32-bit integer	No	-1	How many path levels to descend. -1 indicates unlimited depth which is the default if not set.
crawl_item_timeout	integer	No	600000	Defines the timeout accessing a single file during a crawl. If this timeout is reached, then the crawl job is stopped to prevent the thread from hanging. The default is 600000 milliseconds or 10 minutes.
exclude_paths	list of strings	No	Empty	Regular expression patterns that will not match. If not empty then paths are excluded if any pattern matches.
include_extensions	list of strings	No	Empty	If you would like to index only files with certain extensions that you define. This parameter will only look at the file extension (such as "pdf" or "xml") and not recognize the file's mime type.

Key	Type	Required	Default	Description
include_paths	list of strings	No	Empty	Regular expression patterns to URLs of files. If not empty, the pattern must match to include blank, all sub-directory paths (limited by the crawl_depth).
index_directories	boolean	No	False	By default, directories will not documents. If you would prefer filesystem directories appear in documents, select this option.
max_bytes	long	No	10,485,760	Defines the maximum size of a crawl. Optional, default is -1, per document.
max_docs	integer	No	-1	Defines the maximum number to crawl. A value of -1 will crawl documents (in conjunction with source attributes which may limit).
max_threads	integer	No	1	Defines the maximum number crawler should use. Optional, which is single-threaded.
maximum_connections	integer	No	1000	Limits the number of tasks (connections) will be created. Each document is a task, and each task creates connection to the Windows Share. With a large number of documents create a large number of connections may cause errors or degraded performance.
password	string	Yes	Null	Your Amazon S3 SecretAccessKey.
persisted_URIs_transaction_size	integer	No	1000	Limits the size of transaction to list of persisted URIs to support crawling.
url	string	Yes	Null	For S3 (Hadoop over Amazon) is a fully-qualified URL that starts with protocol, the name of the bucket, and the path inside the bucket. All URI directories that contain files to crawl must end with a trailing slash. New requirement for v2.1.

Key	Type	Required	Default	Description
				Both <code>AccessKeyId</code> and <code>Secret</code> needed: submit <code>AccessKeyId</code> username and <code>SecretAccessK</code> password. You can also pass t as part of the URL in the follow s3://<username>@<password>. However, Amazon S3 creden contain characters that are no URLs. In that case, you must credentials by setting the use: { {password properties explicit
username	string	Yes	Null	Your Amazon S3 AccessKeyId
verify_access	boolean	No	True	By default, LucidWorks Search verify the data source is acces time. To be able to create a d without verifying access, chan false . Note, however, that if L Search cannot access the data not be able to crawl it.

- ✔ The `include_paths` and `exclude_paths` attributes for all Remote File System data sources use [Java Regular Expressions](#).

Example S3H data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "tree",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "exclude_paths": [],
  "id": "372492c8cf524905a25b99f63c589933",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
```

```
"indexing": true,
"mapping": {
...
},
"max_bytes": 10485760,
"max_docs": -1,
"max_threads": 1,
"maximum_connections": 1000,
"name": "HDFS on S3",
"output_args": "threads=2,buffer=1",
"output_type": "solr",
"parsing": true,
"password": "AWS-SecretKey",
"persisted_URIs_transaction_size": 1000,
"type": "s3h",
"url": "s3:/hdfs/test1/",
"username": "AWS-AccessKey",
"verify_access": true
}
```

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

🚩 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.

Key	Description
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

HDFS Data Sources

LucidWorks Search has been tested with Hadoop v1 and v2. The HDFS data source can use the version of Hadoop bundled with LucidWorks Search (v2.0.5-alpha), or an external Hadoop of either version 1 or version 2 can be used by specifying a parameter in the system `master.conf`. The parameter varies depending on the Hadoop version being used: enter `-Dhadoop` if using Hadoop v1 or `-Dhadoop2` for Hadoop v2. The value entered should be the path to the Hadoop client libraries, and complete (non-relative) paths should be used (such as `-Dhadoop2=/path/to/Hadoop`). Note that this system parameter will apply to all HDFS data sources.

As of v2.7, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

This data source differs from the various Hadoop data sources in that this data source type is designed to be polite and crawl through the content stored in HDFS just as if it crawled a web site or any other file system. The distribution-specific Hadoop data sources, however, designed to take full advantage of the scaling abilities of Hadoop's MapReduce architecture.

- [HDFS Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [HDFS Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

HDFS Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an HDFS data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the LWE-Core component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. These are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the cluster and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent requests to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 2, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must not the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lo script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested

Key	Type	Required	Default	Description
				processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

HDFS Type-Specific Attributes

When creating a data source of type **hdfs**, the value **lucid.fs** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	If true , documents that failed processing (due to invalid for parsing errors, IO failures, etc) added to the index with whatever metadata that could be retrieved the document (if it was only partially parsed, for example) and the error in the "parse" or "failed" which would be added to the document.
bounds	string	No	None	Either tree to limit the crawl to a subtree, or "none" for no limit. For example, if crawling <code>hdfs://10.0.0.50/docs</code> , the option is the equivalent of <code>hdfs://10.0.0.50/docs/*</code> and <code>lucid.fs</code> would not crawl <code>hdfs://10.0.0.50/other</code> . If you require more advanced crawling options, you should choose none and use the <code>includes</code> or <code>excludes</code> options.
converter	string	No	Null	Multi-record Hadoop files, such as SequenceFiles and MapReduce files, can be crawled with a sub-crawler to produce valid Solr documents (as there are Hadoop records in each fileset). The keys and values must be converted to SolrInputDocuments in order to be indexed. This can be done in one of two ways: by using one of the subclasses of <code>com.lucid.crawl.fs.hdfs.C</code>

Key	Type	Required	Default	Description
				and specified by setting the <code>converter</code> attribute to a fully class name that extends <code>Converter</code> . If no value is specified, then a <code>DefaultConverter</code> will be used, creates a single document per key-value pair and invokes the <code>toString()</code> method and adds the "body" field.
<code>crawl_depth</code>	32-bit integer	No	-1	How many path levels to descend. '-1' to indicate unlimited depth. also the default if left blank.
<code>crawl_item_timeout</code>	integer	No	600000	Defines the timeout accessing parsing for a single file during a crawl. If a file exceeds this timeout, the crawl job will be stopped to prevent a thread hanging for no reason. default is 600000 millisecond minutes.
<code>exclude_paths</code>	list of strings	No	Empty	Regular expression patterns that must not match. If not empty, any path is excluded if any pattern matches the URL.
<code>include_extensions</code>	list of strings	No	Empty	If you would like to index only specific file extensions that you list them here. This parameter looks at the file extension (such as ".txt" or ".xls"), and will not recognize file's mime types.
<code>include_paths</code>	list of strings	No	Empty	Regular expression patterns that must match on full URLs of files. If not empty, at least one pattern must match to include a file. If left blank, all subdirectory paths will be followed (limited by the <code>crawl_depth</code>). This feature can be used to limit a crawl to specific subdirectories or a specific directory path. For example, if the directory path is <code>/Path/to/Files</code>

Key	Type	Required	Default	Description
				includes could be used to limit to subdirectories /Path/to/Files/Archive/20 /Path/to/Files/Archive/20
index_directories	boolean	No	False	By default, directories will not indexed as documents. If you prefer to have filesystem direc appear in your index as docum select this option.
max_bytes	long	No	10,485,760	Defines the maximum size of document to crawl. Optional, 10,485,760 which is 10Mb per document.
max_docs	integer	No	-1	The maximum number of doc crawl. The default is -1, which collect documents found (in cc with the other data source att which may limit the number o documents).
max_threads	integer	No	1	Defines the maximum number threads the crawler should use. Optional, default is 1, which is single-threaded. The FTP data type does not support a max_ value greater than 1.
maximum_connections	integer	No	1000	Limits the number of tasks (connections) that will be crea document to be crawled is a t each task creates a new conn the Windows Share system. A large number of documents, t create a large number of conn which may cause errors or de performance.
persisted_URIs_transaction_size	integer	No	1000	Limits the size of transaction t the list of persisted URIs to su incremental crawling.
url	string	Yes	Null	

Key	Type	Required	Default	Description
				A fully-qualified Hadoop file system URL, including the protocol (hdfs), name and port of the namenode, and the path of the target resource to crawl. For example: <code>hdfs://namenode:9000/path</code> .
verify_access	boolean	No	True	By default, LucidWorks Search attempts to verify the data source is accessible at create time. To create a data source without verifying access, change this value to <code>false</code> . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.

- ✔ The includes and excludes attributes for all Remote File System data sources use [Java Regular Expressions](#).

Example HDFS data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "tree",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "converter": null,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "exclude_paths": [],
  "id": "1056210f291c4c6faf98643946feb319",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
  "indexing": true,
  "mapping": {
    ...
  },
}
```

```
{
  "max_bytes": 10485760,
  "max_docs": -1,
  "max_threads": 1,
  "maximum_connections": 1000,
  "name": "HDFS",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "parsing": true,
  "persisted_URIs_transaction_size": 1000,
  "type": "hdfs",
  "url": "hdfs://hdfs/test1/",
  "verify_access": true
}
```

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

POST `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Intel Distribution for Hadoop Data Sources

The Intel Distribution for Hadoop data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements. LucidWorks Search has been tested with Intel Distribution for Hadoop v3.0.2.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.



Before using the Intel Distribution for Hadoop Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [Intel Distribution for Hadoop Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Intel Distribution for Hadoop Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Intel Distribution for Hadoop Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Intel Distribution for Hadoop data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some

performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. These are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the example and creating the data source for collection1, the http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent requests to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1000.

Key	Type	Required	Default	Description
				<p>which means no additional buffering. In c this value to higher than one has little im performance when the number of threads: 1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the fo <i>name</i>", without any file extension. The system look for the script name with a .js file extensor only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon": output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be

Key	Type	Required	Default	Description
				mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs,</p>

Key	Type	Required	Default	Description
				<p>lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code> Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code> Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code> Any class that ends with <code>*LongField</code> becomes <code>LONG</code> Anything else not listed above becomes <code>STRING</code>
unique_key	string	No	"id"	

Key	Type	Required	Default	Description
				Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the

Key	Type	Required	Default	Description
				final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	

Key	Type	Required	Default	Description
				When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Intel Distribution for Hadoop Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.intel** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for <code>\$HADOOP_HOME/bin/hadoop</code> . If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to <code>\$HADOOP_HOME/bin/hadoop</code> , then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example Intel Distribution for Hadoop data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.intel",
  "hadoop_home": "/hadoop",
```

```

    "id": "0ad83e0772e0497f8efbbed1215e6f48",
    "job_jar": "hadoop-lws-job.jar",
    "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true
-c com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
    "name": "Intel Hadoop",
    "output_args": "threads=2,buffer=1",
    "output_type": "solr",
    "type": "hadoop",
    "verify_access": false
}

```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Required
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/*</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing

Argument	Required	Description
		into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this <code>mapperClass</code> , the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> ,	Sequen

Mapper Class Name	Description	Input
	where the key is any Writable and the value is text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 `PUT /api/collections/collection/datasources/ id`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 `DELETE /api/collections/collection/datasources/ id`

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

MapR Hadoop Data Sources

The MapR Hadoop data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements. LucidWorks Search has been tested with MapR Hadoop v3.0.2.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.



Before using the MapR Hadoop Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [MapR Hadoop Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [MapR Hadoop Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

MapR Hadoop Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an MapR Hadoop data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the directory and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the Zook again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if <code>multiValued=false</code> and multiple values are encountered, only the first value is retained and all other values are discarded.
<code>original_content</code>	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not

Key	Type	Required	Default	Description
				created and documents are not preserved unless as a result of setting other options above.

MapR Hadoop Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.mapr** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for \$HADOOP_HOME/bin/hadoop. If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to \$HADOOP_HOME/bin/hadoop, then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example MapR Hadoop data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.mapr",
  "hadoop_home": "/hadoop",
  "id": "0ad83e0772e0497f8efbbed1215e6f48",
  "job_jar": "hadoop-lws-job.jar",
  "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true"
```

```
-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
  "name": "MapR Hadoop",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "type": "hadoop",
  "verify_access": false
}
```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Req
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/*</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud

Argument	Required	Description
		mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this mapperClass, the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> , where the key is any <code>Writable</code> and the value is	Sequen

Mapper Class Name	Description	Input
	text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls  
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of  
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk  
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

MongoDB Data Sources

The MongoDB crawler allows indexing of a MongoDB instance.

When a MongoDB data source runs for the first time, an initial synchronization with MongoDB is performed to get all content.

Once the initial synchronization is completed, updates and new documents are indexed by reading the oplog and processing the entries. At any time a full synchronization can be performed again if necessary.

- [MongoDB Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [MongoDB Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

MongoDB Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a file data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are:

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. It is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in <code>/opt/lucidworks</code> and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a space with no whitespace between them. For example, <code>threads=2 buffer=1</code>.

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdateCorruptor": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorScriptProcessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

MongoDB Type-Specific Attributes


When creating a data source of type **mongodb**, the value **lucid.mongodb** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
<code>collections</code>	string	No	null	<p>The MongoDB collection to index, in the form <code>databaseName.collectionName</code>. Multiple collections can be specified and separated by commas.</p> <p>Wildcards can be used to specify a range of databases and/or collections. If <code>*.*</code> is defined, all databases and all collections of each database will be crawled assuming the <code>username</code> entered is one which has "admin" access (because the crawler issues a "listDatabases" command to MongoDB, which requires admin-level access). Alternately, a specific database can be specified with a wildcard for collections, such as <code>databaseName.*</code>.</p> <p>If this is left empty, all collections accessible by this crawler will be indexed. If the <code>username</code> and <code>password</code> entered require authentication only to a specific database and collections and <code>collections</code> is left empty, only the collections accessible by that username and password will be indexed.</p>

Key	Type	Required	Default	Description
				Note that the collect referred to here are different from the LucidWorks Search c of collections, and re the structure of the MongoDB instance o
converter_class	string	No	com.lucid. mongodb.converters. DotRepresentation-TwoConverter	Defines the class to convert the MongoDB documents into a fla document for indexi Document structure preserved by transfc the MongoDB nested into "paths" for Solr names. For example nested address in MongoDB would bec address.location. and address.location. The default should b retained unless you implemented a custc converter class and it to the .jar file of tl crawler.
host	string	Yes	localhost	The hostname of the MongoDB instance.
port	integer	Yes	27017	The port of the Mong instance.
perform_initial_sync	boolean	Yes	True	If true , the connect process all data from defined MongoDB collection. This shou true for the initial synchronization. If o updates are needed, change this to false

Key	Type	Required	Default	Description
				you believe you have gotten out of sync with oplog (see the <code>process_oplog</code> option for more detail), then you could do an initial sync again and all the previously crawled data will be replaced with new crawl.
<code>process_oplog</code>	boolean	Yes	True	If true , will watch the MongoDB oplog for read, insert, update, and/or delete operations and process them accordingly. This should be true for regular operation, at initial synchronization. Change this to false if you intend to do a full synchronization with MongoDB collection.
<code>username</code>	string	No	null	The username to use to access the MongoDB instance via a username and password. If the username is used to access a limited set of collections, only those collections will be indexed.
<code>password</code>	string	No	null	The password to use to access the MongoDB instance via a username and password.
<code>verify_access</code>	boolean	No	True	By default, LucidWorks Search will attempt to verify the data source is accessible at creation. To be able to create a source without verifying access, change this to false . Note, however,

Key	Type	Required	Default	Description
				that if LucidWorks Search cannot access the data source, it will not be to crawl it.

 In v2.5.1, the MongoDB data source type did not include the `verify_access` attribute, so does not validate that the MongoDB instance is accessible to the LucidWorks Search crawler during data source creation. This may lead to errors when attempting to crawl if it is not possible for the [LWE-Connectors component](#) to connect to the database with the supplied host, port, username, password and collections combination. In 2.5.2, the `verify_access` attribute has been added to ensure LucidWorks Search can reach the data source at create time.

Example MongoDB data source (without mapping attributes):

```
{
  "category": "Other",
  "collection": "collection1",
  "collections": "",
  "commit_on_finish": true,
  "commit_within": 900000,
  "converter_class": "com.lucid.mongodb.converters.DotRepresentationTwoConverter",
  "crawler": "lucid.mongodb",
  "host": "localhost",
  "id": "30fd65168c0a480f94cc203b926d527d",
  "mapping": {
    ....
  },
  "name": "MongoDB test",
  "output_args": null,
  "output_type": "solr",
  "password": "admin",
  "perform_initial_sync": true,
  "port": 27017,
  "process_oplog": true,
  "type": "mongodb",
  "username": "admin",
  "verify_access": true
}
```


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source



The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the `id`. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Pivotal Hadoop Data Sources

The Pivotal Hadoop data source uses a MapReduce-enabled crawler designed to leverage the scaling qualities of [Apache Hadoop](#) while indexing content into LucidWorks. In conjunction with LucidWorks' usage of SolrCloud, applications should be able to meet their large scale indexing and search requirements. LucidWorks Search has been tested with Pivotal Hadoop v3.0.2.

To achieve this, the Hadoop connector consists of a series of MapReduce-enabled Jobs to convert raw content into documents that can be indexed into LucidWorks, and also relies on MapReduce-ready document conversion via [Apache Tika](#) and writing of documents to LucidWorks.

The definition of the Hadoop job includes several parts, defined with the `job_jar_args` parameter to the API. These job arguments include the location of your Solr instance (standalone) or ZooKeeper ensemble (SolrCloud); the Mapper class and input format that should be used, which is related to the type of content you want to process (Zip files vs. SequenceFiles vs. general documents, etc.); the output format the Hadoop job should produce, as well as other parameters explained below.



Before using the Pivotal Hadoop Data Source type, please review the section [Using the Hadoop Crawlers](#).

- [Pivotal Hadoop Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [MapR Hadoop Type-Specific Attributes](#)
 - [Job Jar Arguments](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Pivotal Hadoop Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Pivotal Hadoop data source. Note that this data source type does not support field mapping, so those attributes are invalid for this data source type.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the directory and creating the data source for collection1, then the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must not the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lo script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after

Key	Type	Required	Default	Description
				some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Required	Default	Description
				<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's <code>multiValued</code> attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the</p>

Key	Type	Required	Default	Description
				<p>expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false in the target schema, then all values are </div>

Key	Type	Required	Default	Description
				<p>concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if <code>multiValued=false</code> and multiple values are encountered, only the first value is retained and all other values are discarded.
<code>original_content</code>	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source</p>

Key	Type	Required	Default	Description
				types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the

Key	Type	Required	Default	Description
				<p>schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified</p>

Key	Type	Required	Default	Description
				and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not

Key	Type	Required	Default	Description
				created and documents are not preserved unless as a result of setting other options above.

MapR Hadoop Type-Specific Attributes

When creating a data source of type **hadoop**, the value **lucid.hadoop.pivotal** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
hadoop_home	string	Yes	Null	The path to a Hadoop installation, where LucidWorks Search will look for <code>\$HADOOP_HOME/bin/hadoop</code> . If LucidWorks Search is not installed on the same server as the Hadoop nameNode, or another node of the cluster that has access to <code>\$HADOOP_HOME/bin/hadoop</code> , then a client can likely be installed on the LucidWorks Search server that is configured to access the Hadoop installation. The path to the files given as the input path in the <code>job_jar_args</code> will be used to find the files to crawl.
job_jar	string	Yes	hadoop-lws-job.jar	If true , the default, metadata extracted from Tika during processing is added to documents. If false , then only minimal data is added, such as the timestamp and URL of the document.
job_jar_args	boolean	Yes	Null	The parameters that will be passed to Hadoop for processing the job. See the section on Job Jar Arguments below.

Example MapR Hadoop data source:

```
{
  "category": "FileSystem",
  "collection": "collection1",
  "crawler": "lucid.hadoop.pivotal",
  "hadoop_home": "/hadoop",
  "id": "0ad83e0772e0497f8efbbed1215e6f48",
  "job_jar": "hadoop-lws-job.jar",
  "job_jar_args": "com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true"
```

```
-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/9CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://10.0.1.7:8888/solr",
  "name": "Pivotal Hadoop",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "type": "hadoop",
  "verify_access": false
}
```

Job Jar Arguments

Hadoop job jar arguments allow you to define the type of content in your Hadoop filesystem and choose "ingest mappers" appropriate for that content. The arguments also allow you to define parameters for the mappers.

The job arguments must conform to the following structure and must be entered in the proper order, as shown below:

1. The main class must be specified. For all of the mappers available, this is always defined as `com.lucid.sda.hadoop.ingest.IngestJob`.
2. System or Mapper-specific arguments, defined as `-Dargument=value`. In many cases, the arguments needed are only needed for certain Mapper class(es) that is defined in later in the argument string.

There are several possible arguments:

Argument	Value Type	Req
<code>-Dlww.commit.on.close</code>	boolean	No
<code>-DcsvDelimiter</code>	string	No
<code>-DcsvFieldMapping</code>	key-value pair	No
<code>-DtikaProcessorClass</code>	string	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.regex</code>	string	No

Argument	Value Type	Required
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.groups_to_fields</code>	key-value pair	No
<code>-Dcom.lucid.sda.hadoop.ingest.RegexIngestMapper.match</code>	boolean	No

Other arguments not defined here can be supplied as needed and they will be added to the Hadoop configuration. These arguments should be defined with the `-Dargument=value` syntax.

- Key-value pair arguments that apply to the ingest job generally. These arguments are expressed as `-argument value`.

There are several possible arguments:

Argument	Required	Description
<code>-cls</code>	Yes	The mapper class. This class must correspond to the content being indexed to ensure proper parsing of documents. See the Mapper Class table below for details of each available mapper.
<code>-c</code>	Yes	The collection name. This is the same collection where you are creating the data source, such as <code>collection1</code> .
<code>-of</code>	Yes	The output format. For all cases, you can use the default <code>com.lucid.sda.hadoop.io.LWMapRedOutputFormat</code> .
<code>-i</code>	Yes	The path to the Hadoop input data. This path should point to the HDFS directory. If the defined location is not a specific filename, the syntax must include a wildcard expression to find documents, such as <code>/data/*</code> .
<code>-s</code>	Not if <code>-zk</code> is used.	The Solr URL. In LucidWorks Search, this would be the URL of the LWE-Core component . In a default installation, this would be http://localhost:8888/solr . Use this parameter if you are indexing into a LucidWorks Search installation that is <i>not</i> running in SolrCloud mode. If LucidWorks Search is running in SolrCloud

Argument	Required	Description
		mode, you should use <code>-zk</code> instead. If not using <code>-s</code> , you should use <code>-zk</code> .
<code>-zk</code>	Not if <code>-s</code> is used.	<p>A list of ZooKeeper hosts, followed by the ZooKeeper root directory. For example, <code>10.0.1.1:2181,10.0.1.2:2181,10.0.1.3:2181/lws</code> would be a valid value.</p> <p>This parameter is used when running LucidWorks Search in SolrCloud mode, and allows the output of the crawl to be routed via ZooKeeper to any available node. If you are not running LucidWorks Search in SolrCloud mode (and don't have ZooKeeper), use the <code>-s</code> argument instead. If not using <code>-zk</code>, you should use <code>-s</code>.</p> <p>If you have installed LucidWorks Search using the instructions at Cluster Installation, you may not have defined the root directory for your ZooKeeper ensemble. In that case, the default is used (<code>"/lws"</code>).</p>
<code>-mt</code>	No	The mimeType of the incoming content. Used with the DirectoryIngestMapper and the ZipIngestMapper. This argument saves a little bit of time in processing if you are sure of the mimeTypes present in your content before crawling. However, Tika can do mimeType identification, so this is not required.
<code>-redcls</code>	No	The class name of a custom IngestReducer, if any. In order for this to be invoked, you must also set <code>-ur</code> to a value higher than 0. If no value is specified, then the default reducer is used, which is <code>com.lucid.sda.hadoop.ingest.IngestReducer</code> .
<code>-ur</code>	No	The number of reducers to use when outputting to the OutputFormat. Depending on the output format and your system resources, you may wish to have Hadoop do a reduce step so the output resource is not overwhelmed. The default is 0 , which is to not use any reducers.

So, the proper order for each element of the argument is as follows:

1. Main ingest class.
2. Mapper arguments, which usually vary depending on the Mapper class chosen, in the format of `-Dargument=value`
3. Ingest arguments, which include the input format and the chosen Mapper class, in the format of `-argument value`

Example arguments are shown below in the section [Example Arguments](#).

Mapper Classes

This table defines the available mapper classes and how they can be used.

Mapper Class Name	Description	Input
<code>com.lucid.sda.hadoop.ingest.BehemothIngestMapper</code>	Index files in Behemoth file format.	Sequen
<code>com.lucid.sda.hadoop.ingest.CSVIngestMapper</code>	Index files in CSV file format. With this mapperClass, the <code>csvFieldMapping</code> parameter must be set when creating the data source (with the argument <code>-DcsvFieldMapping</code>). The delimiter can also be changed from the default (a comma ",") with the <code>-DcsvDelimiter</code> parameter.	TextInp
<code>com.lucid.sda.hadoop.ingest.DirectoryIngestMapper</code>	Index a directory of files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	
<code>com.lucid.sda.hadoop.ingest.RegexIngestMapper</code>	Allows definition of an regular expression that is used on the incoming content.	
<code>com.lucid.sda.hadoop.ingest.SequenceFileIngestMapper</code>	Index a <code>SequenceFile</code> . If the value is "text", the string will be used, otherwise the raw bytes will be written.	Sequen
<code>com.lucid.sda.hadoop.ingest.SolrXMLIngestMapper</code>	Index a file in SolrXML format. The file should be in a <code>SequenceFileInputFormat</code> , where the key is any <code>Writable</code> and the value is	Sequen

Mapper Class Name	Description	Input
	text in SolrXML. This mapper requires that the <code>idField</code> parameter be set when creating the workflow job. This mapper supports overriding the default <code>inputFormat</code> of <code>SequenceFileInputFormat</code> if required.	
<code>com.lucid.sda.hadoop.ingest.WarcIngestMapper</code>	Index web archive (<code>.warc</code>) files in <code>WarcFileInputFormat</code> .	WarcFil
<code>com.lucid.sda.hadoop.ingest.ZipIngestMapper</code>	Index <code>.zip</code> files. Tika will be used to extract content from these files, so file types supported by Tika will be parsed.	

Example Arguments

Index CSV files

To index CSV files, you could use the following arguments:

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -DcsvDelimiter=| -cls
com.lucid.sda.hadoop.ingest.CSVIngestMapper -c collection1 -i /data/CSV -of
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -s http://localhost:8888/solr
```

To explain in more detail, here is a breakdown of each parameter:

- Main Class: `com.lucid.sda.ingest.IngestJob`
- We want to commit the documents when finished: `-Dlww.commit.on.close=true`
- The delimiter is a pipe character (`|`): `-DcsvDelimiter=|`
- We have CSV files, so we should use the CSV Mapper Class: `-cls com.lucid.sda.hadoop.ingest.CSVIngestMapper`
- We want to index the documents to "collection1": `-c collection1`
- The documents are located at this path: `-i /data/CSV`
- We'll use the default output format: `-of com.lucid.sda.hadoop.io.LWMapRedOutputFormat`
- We're not using SolrCloud, so the LucidWorks Solr is found at: `-s http://localhost:8888/solr`

Index a Directory of Files with SolrCloud

```
com.lucid.sda.hadoop.ingest.IngestJob -Dlww.commit.on.close=true -cls  
com.lucid.sda.hadoop.ingest.DirectoryIngestMapper -c collection1 -i /data/files -of  
com.lucid.sda.hadoop.io.LWMapRedOutputFormat -zk  
10.0.1.7:2181,10.0.1.8:2181,10.0.1.9:2181/lws
```

In this example, we have defined the job very similarly to the previous example. We defined that LucidWorks Search should commit the documents when finished, defined the Mapper Class, specified a collection ("collection1"), pointed the crawler to the input directory (/data/files), and defined the output format.

Note that in this case instead of defining the location of Solr, we used the `-zk` parameter to define a list of hosts running our ZooKeeper ensemble. We can list the host:port locations separated by commas, and then finally define the root directory, which in this case is `/lws`, which is the default, but another root directory may have been defined during installation. See also [Cluster Installation](#) for more details on defining the root directory for your ZooKeeper ensemble during LucidWorks Search installation.

While a custom Tika processor is possible with the DirectoryIngestMapper, we aren't defining one, so we don't need to use the `-DtikaProcessor` option.


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Push Data Sources

In LucidWorks Search v2.7, the "external" data source type has been replaced with the "push" connector, which provides a simpler interface for pushing documents into LucidWorks Search. The push connector uses the embedded JettySolrRunner to push the documents.

It has the benefit of using the field mapping functionality of Solr, but can also process adds, deletes, and updates to documents in the same way that Solr can (i.e., using the update requestHandlers for CSV, XML, JSON, etc.). It can also send the output through any of the available output options described in the advanced fields section below. Document counts should also be reflected properly in the Admin UI and data source history APIs.

If you are using a smart SolrJ client already (i.e., CloudSolrServer), it's worth weighing the benefits of this data source against the drawback that it is a single endpoint which may become a bottleneck or single point of failure. However, the ability to use the LucidWorks Search processing chain may still outweigh this disadvantage.

ta source, use the `lucidworks_fields` parameter to add LucidWorks specific data source fields (i.e., `data_source`, `data_source_name` and `data_source_type`) to the documents. This parameter is added in the field mapping definition, described in the section on [common attributes](#). If this option is not selected, it may be difficult to identify the added documents if they need to be deleted later.

More information about how external data sources work is available in the section [Pushing Content to LucidWorks](#).

- [External Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Push Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

External Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a Push data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some

performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. These are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the path and creating the data source for collection1, then http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent requests to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1000.

Key	Type	Required	Default	Description
				<p>which means no additional buffering. In c this value to higher than one has little im performance when the number of threads: 1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the fo <i>name</i>", without any file extension. The system look for the script name with a .js file extensor only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon": output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be

Key	Type	Required	Default	Description
				mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs,</p>

Key	Type	Required	Default	Description
				<p>lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	

Key	Type	Required	Default	Description
				Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the

Key	Type	Required	Default	Description
				final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	

Key	Type	Required	Default	Description
				When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Push Type-Specific Attributes

When creating a data source of type **push**, the value of **lucid.push** must be specified for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
port	integer	Yes	none	The port that will be used to push content to Solr.

Sample Push data source (without mapping attributes)

```
[
  {
    "caching": false,
    "category": "push",
    "collection": "collection1",
    "commit_on_finish": true,
    "commit_within": 900000,
    "crawler": "lucid.push",
    "id": "320017d3e4874cefb7da7673d716d361",
    "indexing": true,
    "mapping": {
      ...
    },
    "name": "Push connector",
    "output_args": "threads=2,buffer=1",
    "output_type": "solr",
    "parsing": true,
    "port": 8425,
    "type": "push",
    "url": "http://10.0.1.7:8425/solr"
  }
]
```


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

SharePoint Data Sources

LucidWorks supports crawling a SharePoint Repository running on the following platforms:

- Microsoft Office SharePoint Server 2007
- Microsoft Windows SharePoint Services 3.0
- Microsoft SharePoint 2010

LucidWorks Search does not support crawling a SharePoint repository running on Microsoft Portal SharePoint Server 2003 or Microsoft Windows SharePoint Services 2.0.

SharePoint data sources will only discover new or changed content, so if recrawling a SharePoint server and no documents are found, it is likely because none of the content was changed from prior crawls.

- [Getting Ready to Index SharePoint Content](#)
- [SharePoint Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [SharePoint Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Getting Ready to Index SharePoint Content

In order to index SharePoint content, Google Services for SharePoint must be installed on the SharePoint server to be able to fully use the embedded APIs that crawl a SharePoint server. The files are included with LucidWorks when it is installed locally and should be moved to the SharePoint server as described below. Customers of LucidWorks Search hosted on AWS or Azure can download the required files from [Google SharePoint Connector downloads](#); look for the latest version of the "Google SharePoint Connector" in either the Binary or Source distributions (not the "Google Search Box").

1. Login to the SharePoint server whose sites are to be crawled by the SharePoint data source.
2. Go to the ISAPI directory of SharePoint. If you are using the standard default installation, the path to this directory would be C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\ISAPI for SharePoint 2007 and C:\Program Files\Common Files\Microsoft Shared\web server extensions\14\ISAPI for SharePoint 2010.
3. Copy the following files from your LucidWorks installation into SharePoint's ISAPI folder specified in previous step:

```
$LWS_HOME/app/webapps/ext/sharepoint_service/Bulk Auth/SharePoint
2007/GSBulkAuthorization.asmx
$LWS_HOME/app/webapps/ext/sharepoint_service/Bulk Auth/SharePoint
```

```

2007/GSBulkAuthorizationdisco.aspx
$LWS_HOME/app/webapps/ext/sharepoint_service/Bulk Auth/SharePoint
2007/GSBulkAuthorizationwsdl.aspx

$LWS_HOME/app/webapps/ext/sharepoint_service/Site Discovery/SharePoint
2007/GSSiteDiscovery.asmx
$LWS_HOME/app/webapps/ext/sharepoint_service/Site Discovery/SharePoint
2007/GSSiteDiscoverydisco.aspx
$LWS_HOME/app/webapps/ext/sharepoint_service/Site Discovery/SharePoint
2007/GSSiteDiscoverywsdl.aspx

$LWS_HOME/app/webapps/ext/sharepoint_service/Acl/GssAcl.asmx
$LWS_HOME/app/webapps/ext/sharepoint_service/Acl/GssAcldisco.aspx
$LWS_HOME/app/webapps/ext/sharepoint_service/Acl/GssAclwsdl.aspx

```

SharePoint Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a SharePoint data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are: <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS)

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>.

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • com.lucid.sda.hbase.lws.HBaseUpdateCon output will be sent to an HBase implementation <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the example and creating the data source for collection1, the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen with <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a comma with no whitespace between them. For example: <code>"output_args": "buffer=2,threads=10"</code> (and will use the default Solr location). If either value is missing, the default is used. • <code>output_type</code> is "com.lucid.crawl.impl.FileUpdate": The <code>output_args</code> must be a URI string for a file path pointing to either a directory (which must exist) or a file (which will be created during the crawl (which must not be the crawl)). The path will be interpreted as entered.

Key	Type	Required	Default	Description
				<p>paths should be used whenever possible. Relative paths are interpreted relative to the working directory of the component, which is <code>\$LWS_HOME</code>.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "<code>com.lucid.crawl.script.ScriptPreprocessor</code>" The <code>output_args</code> are a script name, in the form of "<code>name</code>", without any file extension. The system will look for the script name with a <code>.js</code> file extension. Only Javascripts are allowed at this time. The location of the script must be <code>\$LWS_HOME/conf/data_source</code> • <code>output_type</code> is "<code>com.lucid.sda.hbase.lws.HBaseUpdateConnector</code>" <code>output_args</code> must be the host:port of the ZooKeeper, again used with LucidWorks Big Data only. A value is interpreted by HBase, and will be similar to local <p>If using a custom implementation of <code>UpdateConnector</code>, the <code>output_type</code> attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.

Key	Type	Required	Default	Description
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the

Key	Type	Required	Default	Description
				documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked.

Key	Type	Required	Default	Description
				<p>If the source field name exists in the schema, it will be indexed to that field.</p> <ol style="list-style-type: none"> 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's <code>multiValued</code> attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or</p> </div>

Key	Type	Required	Default	Description
				<p>default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.

Key	Type	Required	Default	Description
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <div> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p> </div>
types	JSON string-string	No	"date": "DATE", "datecreated":	A map pre-initialized from the current schema. Additional validation can be performed on

Key	Type	Required	Default	Description
			"DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes DATE* Any class that ends with <code>*DoubleField</code> becomes DOUBLE Any class that ends with <code>*FloatField</code> becomes FLOAT Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes INT Any class that ends with <code>*LongField</code> becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	<p>Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If</p>

Key	Type	Required	Default	Description
				performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

SharePoint Type-Specific Attributes

The SharePoint crawler will index all content in the repository, including public and personal sites, files, discussion boards, calendars, contacts, and images. When creating a data source of type **sharepoint**, the value of **lucid.gcm** must be used for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	If true , documents that failed to parse, parsing errors, IO failures, etc. will be indexed with whatever metadata from the document (if it was onl

Key	Type	Required	Default	Description
				example) and the reason for the "fetch" field, which would be added to the log.
aliases	map of string to string	No	Null	Allows mapping of source URL paths to new paths. The paths are used to rewrite URLs before indexing.
domain	string	Yes	Empty	The domain where the username is located.
enable_security_trimming	boolean	No	False	This enables restriction of search results to the person submitting the query.
excluded_urls	string	No	Empty	Directories on the server that should be excluded from the crawl. The regular expression syntax can be used. The excluded_urls as is used for including/excluding files.
feed_unpublished_documents	boolean	No	True	If there are unpublished documents in the repository, you can still crawl and index them by selecting this option. The documents are subject to the document access control rules, the same as published documents.
groupname_format_in_ace	string	No	domain groupname	Defines how to store the access control entries (ACEs) in the ACL. ACEs can be stored as tuples (domain, groupname) or as a single string with the group domain added as a prefix.
included_urls	string	No	Empty	The directories on the server that should be included in the indexing. If left blank, all paths will be indexed. If you want to restrict the crawling to a specific site, repeat the path with a regular expression to indicate the site. If defining any regular expressions, the regular expressions must be entered with the appropriate escaping.
				The SharePoint data source uses regular expressions, which may be different from the regular expressions used for WebCrawler sources. More information on the regular expressions used by the GNU regular expression documentation .
kdcserver	string	No	Empty	Kerberos KDC Hostname.
ldap_auth_type	string	No	Simple	

Key	Type	Required	Default	Description
				The bind type to use when connecting to the LDAP server. The options are "none", "simple", "ssl", and "tls".
ldap_cache_groups_membership	boolean	No	True	If true , LucidWorks Search will cache group information on the first lookup. This provides faster access if it is needed again.
ldap_cache_refresh_interval	integer	No	7200	How often to refresh the group membership cache in seconds. The default is 7200 seconds.
ldap_cache_size	integer	No	1000	The size of the group membership cache in terms of number of items to store.
ldap_read_groups_type	string	No	TOKEN_GROUPS	The mode to use for reading group information from the LDAP server. There are three possible modes: <ul style="list-style-type: none"> • TOKEN_GROUPS: The fastest mode. The token_group is a computed list of SIDs. • IN_CHAIN: A matching rule chain of user to group until a group is found. • RECURSIVE: The slowest mode. It uses a recursive expansion.
ldap_search_base	string	No	Null	The base node for LDAP user and group searches.
ldap_server_host_address	string	No	Null	The hostname of the LDAP server.
ldap_server_port_number	integer	No	389	The port of the LDAP server.
ldap_server_use_ssl	boolean	No	False	If the connection to the LDAP server should use an SSL connection, enable this option.
max_docs	integer	No	-1	The number of documents to crawl. If set to -1, all documents found by the crawler will be indexed (in conjunction with the other document limits which may limit the crawl).
my_site_base_url	string	No	Null	Used for MOSS 2007 only. The MySite URL is used to determine the complete MySite URL. For example, if the URL http://server.domain/personal/abc would be entered as http://server.domain/personal/abc , the MySite URL and credentials provided will allow the crawler to complete the MySite URL and crawl the site.

Key	Type	Required	Default	Description
password	string	Yes	Null	Password for the username above
push_acls	boolean	No	False	This saves the access control list (ACL) for each document as a field of each document (field). If you are interested in trying to hide the identity of the searcher, you can enable this option. Enabling this without enabling the ACLs is a helpful way to troubleshoot
sharepoint_url	string	Yes	Null	The fully qualified URL for the SharePoint site
tags_file	string	No	Null	<p>This allows control over the metadata that is indexed by LucidWorks Search. The file is the location and name of a JavaScript file. The file includes a list of mimetypes and specifies whether to include or exclude from indexing anywhere accessible to the LucidWorks application, but the best place to put it is in the \$LWS_HOME/conf dir so all configuration files are in one place.</p> <p>The syntax for this file is:</p> <pre> { mimetype1 : {"include_tags": ["tag1", "tag2..."]}, mimetype2 : {"exclude_tags": ["tag1", "tag2..."]} }</pre> <p>Examples of mimetypes include "application/ms-infopath.xml". If you want to include all mimetypes, you should include the "text" variations. It is not possible to include and exclude statements for the same mimetype.</p> <p>If a tag named in this file has descendants will be indexed or not based on the rule of the parent. If you have a tag included, any tags not defined in the file will be indexed.</p>
use_checksum_detection	boolean	No	false	This will allow the crawler to calculate checksums for already visited documents to calculate if they have changed.

Key	Type	Required	Default	Description
				changes have been made to the crawl session.
use_sp_search_visibility	boolean	No	True	When true (default) then SharePoint options will be respected. Google must be installed to use this feature.
username	string	Yes	Null	Username with authorization to repository.
username_format_in_ace	string	No	username	Defines how to store the access individual users. ACEs can be stored with username, or with the user domain.
verify_access	boolean	No	True	By default, LucidWorks Search v data source is accessible at creation. To create a data source without verifying access, set this value to false . Note, however, that LucidWorks Search cannot access the data source to crawl it.
visits_per_url	integer	No	10	If you have a set of URLs whose content changes very frequently (like, even if the crawler may get into an infinite loop re-crawl the same documents even if the content has not changed. This parameter defines the maximum number of times the crawler will visit the URL during the single crawler run. In most circumstances, it won't be necessary to change this value, as the documents won't change often enough to reach this maximum.



The included_urls and excluded_urls attributes for SharePoint data sources use [GNU Regular Expressions](#).

Example SharePoint data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "aliases": null,
  "caching": false,
  "category": "GCM",
  "collection": "collection1",
  "commit_on_finish": true,
```

```
"commit_within": 900000,
"crawler": "lucid.gcm",
"domain": "Corp",
"enable_security_trimming": false,
"excluded_urls": null,
"feed_unpublished_documents": true,
"groupname_format_in_ace": "domain\\groupname",
"id": "ab8f8dacdc4bd8a608c7a0d9537648",
"included_urls": null,
"indexing": true,
"kdcserver": "",
"ldap_auth_type": "Simple",
"ldap_cache_groups_membership": true,
"ldap_cache_refresh_interval": 7200,
"ldap_cache_size": 1000,
"ldap_read_groups_type": "TOKEN_GROUPS",
"ldap_search_base": "",
"ldap_server_host_address": "",
"ldap_server_port_number": 389,
"ldap_server_use_ssl": false,
"mapping": {
...
},
"max_docs": -1,
"my_site_base_url": null,
"name": "SharePoint",
"output_args": null,
"output_type": "solr",
"parsing": true,
"password": "password",
"push_acls": false,
"sharepoint_url": "http://test.example.com/",
"tags_file": null,
"type": "sharepoint",
"use_checksum_detection": false,
"use_sp_search_visibility": true,
"username": "user1",
"username_format_in_ace": "username",
"verify_access": true,
"visits_per_url": 10
}
```


Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Simple Filesystem Data Sources

The simple File data source is essentially the same as the Aperture-based Filesystem data source, with the exception that this data source type can use multiple threads while crawling, and does not go through Aperture-based parsing (and parsing can be skipped entirely if saving the crawl as a batch).

As of v2.7, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

The filesystem must be directly accessible to the LucidWorks server, either because it is on the same server or because it is mounted as a local drive.

- [Filesystem Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Filesystem Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Filesystem Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a File data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	The type of this data source. Valid types are:

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the conter to Solr for indexing. Only Javascript is supporte The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateCon output will be sent to an HBase implementation <i>conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualifie custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be se are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are p defined, <code>output_args</code> will default to the Solr ins in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in th and creating the data source for collection1, th http://127.0.0.1:8888/solr/collection1). Two a parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurr use for sending updates. This does not de use while crawling a data source, but thr updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documen before sending to SolrJ in bulk, which car reduce the number of calls to Solr. The d which means no additional buffering. In c this value to higher than one has little im performance when the number of threads 1; the performance benefits are usually s <code>threads=1</code>, but at the cost of increased J consumption. • When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessorImpl": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper used with LucidWorks Big Data only. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector, the attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Filesystem Type-Specific Attributes

Note however, that the data source `type file` can also be used for a local filesystem. If intending to crawl a remote filesystem, the value **lucid.fs** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
<code>add_failed_docs</code>	boolean	No	False	If true , documents that failed processing (due to invalid formats, parsing errors, IO failures, etc.) will be added to index with whatever metadata that could be retrieved from the document (if it was only partially parsed, for example) and the reason for the error in the "path" or "fetch" field, which would be added to the document.
<code>bounds</code>	string	No	Tree	Either tree to limit the crawl to a strict subtree, or "none" for no limits. For example, if crawling <code>/path/to/files/docs</code> , the tree option is the equivalent of <code>path/to/files/docs*</code> and the none option would mean that <code>lucid.fs</code> would not crawl <code>/path/to/files/other</code> . If you require more advanced crawl limiting, you should choose none and using the includes or excludes options.
<code>crawl_depth</code>	32-bit integer	No	-1	How many path levels to descend. Use '-1' to indicate unlimited depth which is also the default if left blank.
<code>crawl_item_timeout</code>	integer	No	600000	Defines the timeout accessing/parsing for a single file during crawl. If a file exceeds this timeout, then the crawl job will be stopped to prevent the thread hanging for no reason. The default is 600000 milliseconds or 10 minutes.

Key	Type	Required	Default	Description
exclude_paths	list of strings	No	Empty	Regular expression patterns to exclude. URLs must not match. If not empty then a file is excluded if any pattern matches its URL.
include_extensions	list of strings	No	Empty	If you would like to index only a list of specific file extensions that you define, list them here. This parameter will only look at the extension (such as "pdf" or "x") and will not recognize the file's mime types.
include_paths	list of strings	No	Empty	Regular expression patterns to match on full URLs of files. If not empty then at least one pattern must match to include a file. If left blank, all subdirectory patterns will be followed (limited by the crawl_depth). This feature can be used to limit a filesystem crawl to specific subdirectories of a base directory path. For example, if the base directory path is /Path/to/Files, includes could be used to limit the crawl to subdirectories /Path/to/Files/Archive/2010 and /Path/to/Files/Archive/2011.
index_directories	boolean	No	False	By default, directories will not be indexed as documents. If you would prefer to have filesystem directories appear in your index as documents, select this option.
max_bytes	long	No	10,485,760	Defines the maximum size of a document to crawl. Optional, default is -1, which is 10Mb per document.
max_docs	integer	No	-1	Defines the maximum number of documents to crawl. The default is -1.

Key	Type	Required	Default	Description
				-1, which will collect all documents found (in conjunct with other data source attribu
max_threads	integer	No	1	Defines the maximum number threads the crawler should use. Optional, default is 1, which is single-threaded. The FTP data source type does not support max_threads value greater than 1.
maximum_connections	integer	No	1000	Limits the number of tasks (connections) that will be created. Each document to be crawled is a task, and each task creates a connection to the Windows Shell system. With a large number of documents, this could create a large number of connections which may cause errors or degraded performance.
path	string	Yes	Null	The path to the file or directory files that should be indexed.
persisted_URIs_transaction_size	integer	No	1000	Limits the size of transaction to store the list of persisted URIs to support incremental crawling.
remove_old_docs	boolean	No	True	Instructs LucidWorks Search to remove any documents found in subsequent crawls as deleted and remove them from the index. This option is enabled by default.
url	string	No	Null	The URL is constructed from the entry to the path. It does not need to be defined during data source creation.
verify_access	boolean	No	True	By default, LucidWorks Search attempts to verify the data source is accessible at create time. To be able to create a data source

Key	Type	Required	Default	Description
				without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.

- ✔ The `include_paths` and `exclude_paths` attributes for all Remote File System data sources use [Java Regular Expressions](#).

Example file system data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "bounds": "tree",
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "exclude_paths": [],
  "id": "b1d25ec51d094c16835d29ec175f3d7d",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
  "indexing": true,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "max_threads": 1,
  "maximum_connections": 1000,
  "name": "Local filesystem",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "parsing": true,
  "path": "/files/testing",
  "persisted_URIs_transaction_size": 1000,
  "type": "file",
  "url": "file:/files/testing",
}
```



```
"verify_access": true
}
```

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the `id`. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE `/api/collections/collection/datasources/ id`

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

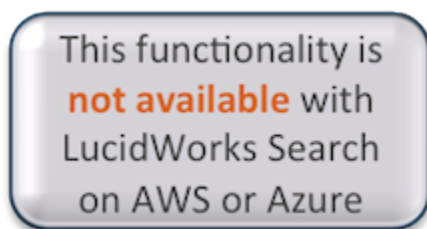
None

Output

Output Content

None

SolrXML Data Sources



The Solr XML data source can only be used on files are formatted according to Solr's XML structure and cannot be used on XML files formatted for another XML standard. Per the Solr standard, all XML files must include the <add> tag in order for the documents to be added to the LucidWorks index. More information on properly formatting a Solr XML file is available at <http://wiki.apache.org/solr/UpdateXmlMessages>. SolrXML data sources are available in **LucidWorks Search on-premise only**.

- [SolrXML Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [SolrXML Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

SolrXML Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an S3 data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.

Key	Type	Required	Default	Description
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. It is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the default location and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a space with no whitespace between them. For example, <code>threads=2 buffer=1</code>.

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon": output_args must be the host:port of the ZooKeeper used with LucidWorks Big Data only. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateCore, the attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

SolrXML Type-Specific Attributes

When creating a data source of type **solrxml**, the value **lucid.solrxml** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
<code>exclude_paths</code>	list of strings	No	Empty	An array of URL patterns to exclude. Used when the data source has been configured to traverse a directory of possible Solr XML files.
<code>generate_unique_key</code>	boolean	No	True	Use true to add a unique identifier to each document if one is not specified already for each document in the file. If not specified, the default is true .
<code>include_paths</code>	list of strings	No	. *.xml	An array of URL patterns to include. Used when the data source has been configured to traverse a directory of possible Solr XML files.
<code>include_datasource_metadata</code>	boolean	No	True	<p>Use true to add <code>data_source</code> and <code>data_source_type</code> fields to each document in addition to fields found in the file. This will allow documents to be included among "Data Source" facets. This has the same effect as the <code>lucidworks_fields</code> parameter defined in the mapping, but <code>include_datasource_metadata</code> will override <code>lucidworks_fields</code> if they are not the same (e.g., if <code>lucidworks_fields</code> is false and <code>include_datasource_metadata</code> is true, the <code>data_source</code> and <code>data_source_type</code> fields will be added to each document).</p> <p>This attribute also has an impact on how existing documents are handled during subsequent crawls. If true, every time a new crawl of</p>

Key	Type	Required	Default	Description
				<p>this data source is started, all documents created by this data source in previous runs will be deleted at the end of the current crawl job (if it fails in the middle, then a mix of old/new documents will co-exist). In this way, LucidWorks will synchronize the documents for this data source with the source file or directory. If false, the deletion step is not performed and all documents found in the source file or directory are treated as new. If the ID of the incoming document matches an existing document, the existing document is replaced with the new document, but otherwise no deletions are performed. This is considered "incremental" crawling, as documents with unique IDs are accumulated across all crawl runs.</p>
max_docs	integer	No	-1	<p>The maximum number of documents to crawl. If the value is -1, then all documents found will be collected (in conjunction with other data source attributes which may limit the crawl).</p>
path	string	Yes	Null	<p>The name of the file to read, or directory containing files to read. Paths should be entered as the complete directory path or they will be interpreted as relative to \$LWS_HOME. On Unix systems, this means the path entered should start at root / level; on Windows, the drive letter and full path should be used (such as C:\path). Various types of relative paths, such as ../ or ~/, are not</p>

Key	Type	Required	Default	Description
				supported. The API will attempt to validate that the path is accessible to the LucidWorks Search server and will return an error if it is not.
url	string	No	Null	Read-only value that shows the absolute path. In many cases this will be identical to the path as entered, but if the path was a shortcut or symbolic link, the url will show the real path to the files.
verify_access	boolean	No	True	By default, LucidWorks Search will attempt to verify the data source is accessible at create time. To be able to create a data source without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.

✔ The include_paths and exclude_paths attributes for Solr XML data sources use [Java Regular Expressions](#).

Example SolrXML data source (without mapping attributes):

```
{
  "caching": false,
  "category": "SolrXml",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawler": "lucid.solrxml",
  "exclude_paths": [],
  "generate_unique_key": true,
  "id": "90519b810b2749cb9a848e2bacfd403f",
  "include_datasource_metadata": true,
  "include_paths": [
    ".*\\.xml"
  ],
  "indexing": true,
```

```
"mapping": {
  ...
},
"max_docs": -1,
"name": "Solr documents",
"output_args": null,
"output_type": "solr",
"parsing": true,
"path": "/Users/cassandra4work/Downloads/apache-solr-4.0.0/example/exampledocs",
"type": "solrxml",
"url":
"file:/Users/cassandra4work/Downloads/apache-solr-4.0.0/example/exampledocs/",
"verify_access": true
}
```


Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Twitter Search Data Sources

The Twitter Search data source type uses Twitter's search API to index tweets that match a specific query. It can use any of the syntax options described in the Twitter search API documentation, [Using the Twitter Search API](#).

This data source uses the `lucid.twitter.search` crawler, which is unique from all of the other available crawlers.

- [Getting Ready to Index Tweets](#)
- [Twitter Search Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Twitter Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Getting Ready to Index Tweets

In order to successfully configure the data source, you must first register your application with Twitter and accept their terms of service. The registration process will provide you with the required OAuth tokens you need to access the search API. To get the tokens, follow these steps:

1. Make sure you have a Twitter account, and go to <http://dev.twitter.com/> and sign in.
2. Choose "Create an App" link and fill out the required details. The callback field can be skipped. Hit "Create Application" to register your application.
3. The next page will contain the Consumer Key and Consumer Secret, which you will need to configure the data source in LucidWorks Search.
4. At the bottom of the same page, choose "Create My Access Token".
5. The next page will contain the Access Token and Token Secret, which you will also need to configure the data source in LucidWorks Search.

While you need a Twitter account to register an application, you do not use your Twitter credentials to configure this data source. Take the Consumer Key, Consumer Secret, Access Token, and Token Secret information and store it where you can access it while configuring the data source.

Twitter Search Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a Twitter data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some

performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor : This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. These are dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the path and creating the data source for collection1, the URL is http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent requests to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1000.

Key	Type	Required	Default	Description
				<p>which means no additional buffering. In c this value to higher than one has little im performance when the number of threads: 1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the fo <i>name</i>", without any file extension. The system look for the script name with a .js file extensor only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon": output_args must be the host:port of the ZooK again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be

Key	Type	Required	Default	Description
				mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs,</p>

Key	Type	Required	Default	Description
				<p>lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	

Key	Type	Required	Default	Description
				Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the

Key	Type	Required	Default	Description
				final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	

Key	Type	Required	Default	Description
				When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Twitter Type-Specific Attributes

When creating a data source of type **twitter_search**, the value **lucid.twitter.search** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
access_token	string	Yes	Null	The access token is provided after registering with Twitter and requesting an access token (see above).
consumer_key	string	Yes	Null	The consumer key is provided after registering with Twitter (see above).
consumer_secret	string	Yes	Null	The consumer secret is provided after registering with Twitter (see above). It should be treated as a password for your registered application.
max_docs	long	No	-1	This limits the number of results returned. The default is "-1", which allows the Twitter API to throttle the number of results it returns as a response.
queries	json map	Yes	Null	The query to perform for the API request. The queries should conform to the syntax options described in the Twitter search API documentation, Using the Twitter Search API .
sleep	integer	Yes	10000	Twitter will occasionally throttle requests, in which case you can configure the data source to wait the requisite amount of time before trying again. The default is 10,000 milliseconds, which should be sufficient for most scenarios.
token_secret	string	Yes	Null	The token key is provided after registering with Twitter and requesting an access token (see above). It should be treated as a password for your API access.

Example twitter_search data source

```
{
  "access_token": "Twitter-AccessToken",
  "caching": false,
  "category": "Other",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "consumer_key": "Twitter-ConsumerKey",
  "consumer_secret": "Twitter-ConsumerSecret",
  "crawler": "lucid.twitter.search",
  "id": "aec56efadeea48deb0b037d305ccb7fa",
  "indexing": true,
  "mapping": {
    ...
  },
  "max_docs": -1,
  "name": "Twitter Search",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "parsing": true,
  "queries": [
    "[\"solr, lucene\"]"
  ],
  "sleep": 10000,
  "token_secret": "Twitter-TokenSecret",
  "type": "twitter_search",
}
```

Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

🚩 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content


None

Delete a Data Source



The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Twitter Stream Data Sources

The Twitter Stream data source type uses Twitter's streaming API to index tweets on a continuous basis.

This data source uses the `lucid.twitter.stream` crawler. Unlike other crawlers, which generally have some kind of defined endpoint (even if that endpoint is after hundreds of thousands or millions of documents), this crawler opens a stream and will not stop until Twitter stops. The Stop Crawl button in the Admin UI (or the [Data Source Jobs](#) API) will allow you to stop the stream if necessary.

This data source does not process deletes. Deletes will be shown in the Admin UI and Data Source History statistics, but these are deleted tweets marked as such from the streaming API - the actual tweets may or may not be in the LucidWorks index (and if they were, the data source does not yet process them).

- [Getting Ready to Index Twitter Streams](#)
- [Twitter Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Twitter Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Getting Ready to Index Twitter Streams

In order to successfully configure a Twitter stream, you must first register your application with Twitter and accept their terms of service. The registration process will provide you with the required OAuth tokens you need to access the stream. To get the tokens, follow these steps:

1. Make sure you have a Twitter account, and go to <http://dev.twitter.com/> and sign in.
2. Choose "Create an App" link and fill out the required details. The callback field can be skipped. Hit "Create Application" to register your application.
3. The next page will contain the Consumer Key and Consumer Secret, which you will need to configure the data source in LucidWorks Search.
4. At the bottom of the same page, choose "Create My Access Token".
5. The next page will contain the Access Token and Token Secret, which you will also need to configure the data source in LucidWorks Search.

While you need a Twitter account to register an application, you do not use your Twitter credentials to configure this data source. Take the Consumer Key, Consumer Secret, Access Token, and Token Secret information and store it where you can access it while configuring the data source.

Twitter Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating a Twitter data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.

Key	Type	Required	Default	Description
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	

Key	Type	Required	Default	Description
				The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
<code>output_type</code>	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: The crawl output will be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content before it is sent to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: The crawl output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only.</i> <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
<code>output_args</code>	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in the

Key	Type	Required	Default	Description
				<p>and creating the data source for collection1, the http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible:</p> <ul style="list-style-type: none"> • "threads": Defines the number of concurrent updates to use for sending updates. This does not decrease performance while crawling a data source, but throughput is reduced when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when threads=1, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a comma with no whitespace between them. For example, "output_args": "buffer=2,threads=10" (and will use the default Solr location). If either value is missing, the default is used. • output_type is "com.lucid.crawl.impl.FileUpdate": The output_args must be a URI string for a file pointing to either a directory (which must exist) or a file that will be created during the crawl (which must not exist at the time of the crawl). The path will be interpreted as an absolute path; relative paths should be used whenever possible. Relative paths are interpreted relative to the working directory of the component, which is \$LWS_HOME. • output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the form of "name", without any file extension. The system will look for the script name with a .js file extension; only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data_source_scripts. • output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": The output_args must be the host:port of the ZooKeeper instance used with LucidWorks Big Data only. A valid path will be interpreted by HBase, and will be similar to local

Key	Type	Required	Default	Description
				If using a custom implementation of <code>UpdateCor</code> attribute can be however you defined its use in

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping API](#). Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e.,

Key	Type	Required	Default	Description
				the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then dynamicField will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to dynamicField_sourceName, after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as data_source and data_source_type) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.

Key	Type	Required	Default	Description
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none">1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field.2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field.3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule.4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field.5. If none of these steps has produced a match, the field will be discarded.

multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div><p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p><ul style="list-style-type: none">• For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded.• If field type is set to DATE in the field mapping, first the</div>
-----------	------------------------	----	---	--

Key	Type	Required	Default	Description
				<p>values are checked for validity and invalid values are discarded. If <code>multiValued=false</code> in the target schema, then only the first remaining value will be retained, and all other values are discarded.</p> <ul style="list-style-type: none"> • If field type is <code>STRING</code>, and <code>multiValued=false</code> in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if <code>multiValued=false</code> and multiple values are encountered, only the first value is retained and all other values are discarded.
<code>original_content</code>	<code>boolean</code>	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of</p>

Key	Type	Required	Default	Description
				<p>field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <div> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p> </div>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with <code>DateField</code> becomes DATE* Any class that ends with <code>*DoubleField</code> becomes DOUBLE

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	<p>Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the unique_key attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the schema.xml value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the schema.xml is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their</p>

Key	Type	Required	Default	Description
				multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Twitter Type-Specific Attributes

When creating a data source of type **twitter_stream**, the value **lucid.twitter.stream** must be supplied for the **crawler** attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
access_token	string	Yes	Null	The access token is provided after registering and requesting an access token (see a
consumer_key	string	Yes	Null	The consumer key is provided after registering (see above).
consumer_secret	string	Yes	Null	The consumer secret is provided after registering (see above). It should be treated as a registered application.
filter_follow	list	No	Null	A set of specific Twitter user IDs to filter. Combined with another filter, they act on the stream (i.e., the tweet must match keyword or the location). Note that this can be a screen name, but a numeric ID associated with the ID, you could do an API request like https://api.twitter.com/1/users/show.json?user_id=1234567890 , replacing "usaa" with the user handle. The ID is found in the "id" field of the XML
filter_locations	list	No	Null	A set of bounding boxes (latitude/longitude, left, etc.) to filter the stream for geographic location. Combined with another filter, they act on

Key	Type	Required	Default	Description
				the stream (i.e., the tweet must match keyword or the location).
filter_track	list	No	Null	A set of keywords to filter the stream. another filter, they act as OR statemer the tweet must match the user ID or tl location).
max_docs	long	No	-1	While testing the feed, it may be desir streams to a specific number of tweets "-1", which doesn't close the connectio closed.
sleep	integer	Yes	10000	Twitter will occasionally throttle stream can configure the data source to wait t time before trying again. The default is which should be sufficient for most sce
token_secret	string	Yes	Null	The token key is provided after registe requesting an access token (see above as a password for your API access.
url	string	No	stream.twitter.com	It's recommended to use the default.

Example twitter_stream data source

```
{
  "access_token": "Twitter-AccessToken",
  "caching": false,
  "category": "Other",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "consumer_key": "Twitter-ConsumerKey",
  "consumer_secret": "Twitter-ConsumerSecret",
  "crawler": "lucid.twitter.stream",
  "filter_follow": [
    22072684
  ],
  "filter_locations": null,
  "filter_track": null,
  "id": "aec56efadeea48deb0b037d305ccb7fa",
  "indexing": true,
  "mapping": {
    ...
  },
  "max_docs": 100,
```



```
{
  "name": "Twitter",
  "output_args": "http://127.0.0.1:8888/solr/collection1",
  "output_type": "solr",
  "parsing": true,
  "sleep": 10000,
  "token_secret": "Twitter-TokenSecret",
  "type": "twitter_stream",
  "url": "https://stream.twitter.com"
}
```


Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.

Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id



This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content

JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output

Output Content

None

Delete a Data Source



The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```



DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Web Data Sources

The Web data source allows LucidWorks to crawl HTML pages over the internet or intranet. The Web crawler uses Aperture, an open source crawler, which is intended as a small-scale crawler not designed as a large-scale internet crawler (for example, it is single-threaded, which means a long list of websites may take a long time for a crawl to complete).

Aperture extracts fields from the HTML of a crawled page and passes the extracted data to LucidWorks for further processing, including any field mapping that has been defined. As of LucidWorks v2.1, the Web data source is able to extract all fields from the <META> section of an HTML page (prior versions were only able to extract the "author", "description", and "keywords" fields). These custom <META> fields are added to the index with a "meta_" prefix before the tag name (i.e., a custom field of "date" would be inserted in the index as "meta_date"). If you wish to map these fields to another field, use the "meta_*" field name; if you do not map them, they will be added to the index as "attr_meta_*". This is due to a default [dynamic rule](#) that adds "attr_" to any field that does not exist in the `schema.xml` for the collection; if you have removed this rule, the fields would be indexed as "meta_*".

If accessing the website is only possible through a proxy server, use the `proxy_*` attributes to configure access.

- [Web Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Web Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Web Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an Web data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexing • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. The output is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance in <code>master.conf</code> and the collection that uses the instance. For example, if LucidWorks has been installed in the default location and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 2, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is low.

Key	Type	Required	Default	Description
				<p>1; the performance benefits are usually s threads=1, but at the cost of increased J consumption.</p> <ul style="list-style-type: none"> When using "threads" and "buffer" in con express them as key=value pairs, separa with no whitespace between them. For ex "output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used. output_type is "com.lucid.crawl.impl.FileUp ": The output_args must be a URI string for a f point to either a directory (which must exist) or will be created during the crawl (which must nc the crawl). The path will be interpreted as ente paths should be used whenever possible. Relati interpreted relative to the working directory of component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor ": The output_args are a script name, in the fo name", without any file extension. The system look for the script name with a .js file extensio only Javascripts are allowed at this time. The lc script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateCon output_args must be the host:port of the Zook again <i>used with LucidWorks Big Data only</i>. A va interpreted by HBase, and will be similar to loc <p>If using a custom implementation of UpdateCor attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to

Key	Type	Required	Default	Description
				dynamicField_sourceName, after some cleanup of the source name (non-letter characters are replaced with underscore).
literals	JSON string-string	No	Null	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as data_source and data_source_type) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the</p>

Key	Type	Required	Default	Description
				<p>schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false,	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it</p>

Key	Type	Required	Default	Description
			"mimeType": false, "title": false	<p>may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false </div>

Key	Type	Required	Default	Description
				<p>in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p>

Key	Type	Required	Default	Description
				<p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.</p>
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING

unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources , this parameter would map a field from the incoming documents to be the unique key for all documents.
verify_schema	boolean	No	"true"	If true , the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also

Key	Type	Required	Default	Description
				be more difficult to learn what the final field mapping was. If this value is false , then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	

Key	Type	Required	Default	Description
				When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Web Type-Specific Attributes

When creating a data source of type **web**, the value **lucid.aperture** must be supplied for the **crawler** attribute, described in the section on [common attributes](#).

Key	Type	Required	Default	Description
add_failed_docs	boolean	No	False	<p>If true, documents that failed processing (due to invalid formats, parsing errors, IO failures, etc.) will be added to the index with whatever metadata that could be retrieved from the document (if it was only partially parsed, for example) and the reason for the error in the "parse" or "fetch" field, which would be added to the document. The default for this attribute is false.</p> <p>If this option is enabled after an initial crawl has been run, it is possible that failed documents will not be added. The reason this happens is because this crawler stores documents it has seen before (also known as persistent crawl data or crawl history) and generally skips documents seen on previous crawls if they are unchanged. To force the crawler to try to process a document again, either clear the crawl history (which will cause all documents to be considered "new" again) or update the document so it appears to the crawler to have been changed.</p>
auth	JSON map	No	Empty	Authentication details, if required. LucidWorks first tries to access the site

Key	Type	Required	Default	Description
				<p>without authentication. If a "401 Authentication Required" Error is found, the authentication method (Basic, Digest, NTLMv1 or NTLMv2 only) is selected automatically and the closest authentication tuple is selected from the ones configured for the data source. The values for this attribute are provided in a JSON map (multiples are allowed); each map has the following properties:</p> <ul style="list-style-type: none"> • <code>host</code>: The hostname (or host:port) where this authentication should be used; may be null to indicate any host. • <code>realm</code>: The HTTP realm where this authentication should be used; may be null to indicate any realm. • <code>username</code>: The username; can not be null or empty. • <code>password</code>: The password; can not be null or empty.
bounds	string	No	Tree	<p>Either tree to limit the crawl to a strict subtree, or none for no limits. If you choose tree, the crawler will only access pages using the seed URL as the base path. For example, if crawling http://www.cnn.com/US, the subtree option is the equivalent of http://www.cnn.com/US* and would not crawl http://www.cnn.com/WORLD, http://www.cnnmexico.com/, or http://us.cnn.com/. If you require more advanced crawl limiting, you should choose none and use the <code>include_paths</code> or <code>exclude_paths</code> options.</p>
crawl_depth	32-bit integer	No	3	<p>The maximum number of crawl cycles (hops) from the starting URL to be crawled. Use '0' to indicate only the seed URL, or any number higher than 0 to go deeper into a site. Use '-1' to indicate</p>

Key	Type	Required	Default	Description
				unlimited depth, which is also the default if left empty. Unlimited depth will crawl everything linked from the base URL and linked to those links, even if it is 10 or more levels away. If the base URL is a public internet site, unless you constrain the crawl to the subtree or define Allow/Disallow Paths, the crawler may run forever. Note that the lucid.aperture crawler is not designed to create an index of the entire internet, and there may be severe performance or index space problems if you do not constrain the crawl.
exclude_paths	list of strings	No	Empty	Regular expression patterns that URLs may not match. If not empty then a file is excluded if any pattern matches its URL. This attribute accepts Java regular expressions .
fail_unsupported_file_types	boolean	No	False	If true , documents that cannot be parsed (either because of unspecified errors or because of an unknown file format) will produce an error in the logs. The default behavior is to not report these documents as failures to the log. If <code>add_failed_docs</code> is also checked, the result of these failures will also be added to the index with whatever metadata could be extracted from the content. This is different from <code>warn_unknown_mime_types</code> in the sense that this parameter only applies to content that fails parsing and not content that doesn't have a defined mime type.
ignore_robots	boolean	No	False	If false , the default, the crawler will respect a robots.txt file found at a site, which is a way for site owners to request no crawling of parts or all of their site. This should only be changed to true if the site is known or if the site owner has granted

Key	Type	Required	Default	Description
				permission to ignore it. The crawler obeys most of the robots.txt standard , with the exception of the Crawl-Delay directive.
include_paths	list of strings	No	Empty	<p>Regular expression patterns to match on full URLs of files. If not empty then at least one pattern must match to include a file. If you leave this field empty, all paths will be followed (except when tree is chosen as a bounds parameter), even if they lead away from the original URL entered. To limit crawling to a specific site, repeat the URL with a regular expression to indicate all pages from the site (such as, enter <code>http://www\..lucidworks\.com/.*</code> if you want to crawl all pages under the URL <code>http://www.lucidworks.com</code>).</p> <p>This attribute accepts Java regular expressions.</p>
log_extra_detail	boolean	No	False	<p>If false, the default, LucidWorks limits the amount of information that is printed to the logs about crawl activities to reduce the size of the log files. However, this information can be helpful if documents aren't being indexed as expected. Change to true for more information to be printed to the data/logs/core.<YYYY_MM_DD>.log. The default is for one line to be printed to the log showing the status of the document accessed ("New", "Update", "Delete" or "Unchanged"). When this setting is enabled, two lines will be shown in the logs for every document: one to say the file is being accessed, and a second to show the status of the document. If there is an error in accessing the document, the first line of the extra log detail may be helpful in figuring out the problem.</p>
max_bytes	long	No	10,485,760	Defines the maximum size of any crawled file. The default is -1, which is 10Mb per document.

Key	Type	Required	Default	Description
max_docs	long	No	-1	Defines the maximum number of documents to crawl. The default is -1, which is all found documents, in accordance with other parameters for the data source.
max_retries	integer	No	3	<p>Defines the maximum number of times to retry connecting to the base URL (defined with the <code>url</code> attribute) of a data source in case of a connection failure, such as one caused by a timeout because the system is temporarily down. If the <code>max_retries</code> number is reached, a detailed error is logged; on the next attempt, documents from the current crawl and from previous crawls, will be removed from the index as though they were deleted documents. If the <code>max_retries</code> has not been reached, and a subsequent attempt to crawl the data source is successful, then the counter is reset to 0. A few additional details to note about this behavior:</p> <ul style="list-style-type: none"> documents are only removed if it is the base URL that is no longer accessible. if creating the data source and the base URL is not accessible, the <code>verify_access</code> attribute still applies. if attempts to access the base URL return with a 404 Not Found error, the document will still be considered removed, and all associated documents will also be removed.
proxy_host	string	No	Empty	<p>The host name of the proxy. LucidWorks supports either open or authenticated proxies. If null or absent then defaults configured in <code>conf/lwe-core/defaults.yml</code> are used (see Configuring Default Settings for more information). If no system-wide defaults are defined, then direct access is assumed,</p>

Key	Type	Required	Default	Description
				and all other proxy-related parameters are ignored.
proxy_port	string	No	-1	The port number of the proxy.
proxy_username	string	No	Empty	optional username credential for the proxy.
proxy_password	string	No	Empty	optional password credential for the proxy.
url	string	Yes	Null	The URL that serves as the crawl seed. This is expected to be unique for each data source of this type, which means that creating two data sources for the same seed URL is not possible. Note that the lucid.aperture crawler may not always be able to work successfully with HTTP redirects and may fail on an initial attempt to crawl a site that redirects a user to a different address. In most cases, a second attempt to crawl the data source will be successful.
verify_access	boolean	No	True	By default, LucidWorks Search will attempt to verify the data source is accessible at create time. To be able to create a data source without verifying access, change this value to false . Note, however, that if LucidWorks Search cannot access the data source, it will not be able to crawl it.
warn_unknown_mime_types	boolean	No	False	If true , documents with no mime type specified in the format produce a warning in the log. If the file cannot be processed as plain text, it will be skipped and a warning message will be printed to the log. The default behavior is to skip these documents and not report warnings in the log. This parameter differs from <code>fail_unsupported_file_types</code> in the sense that it only applies to content that does not have a defined mime type.

Example Web data source (without mapping attributes):

```
{
  "add_failed_docs": false,
  "auth": [],
  "bounds": "tree",
  "caching": false,
  "category": "Web",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": 3,
  "crawler": "lucid.aperture",
  "exclude_paths": [
    ".*?showChildren=true",
    ".*?showChildren=false"
  ],
  "fail_unsupported_file_types": false,
  "id": "9b787cd16e354498ae97adefe0817029",
  "ignore_robots": false,
  "include_paths": [
    "http://docs.lucidworks.com/display/help/*.\"",
    "http://docs.lucidworks.com/display/lweug/*.\""
  ],
  "indexing": true,
  "log_extra_detail": false,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "max_retries": 3,
  "name": "LucidWorks Documentation",
  "output_args": null,
  "output_type": "solr",
  "parsing": true,
  "proxy_host": "",
  "proxy_password": "admin",
  "proxy_port": -1,
  "proxy_username": "admin",
  "type": "web",
  "url": "http://docs.lucidworks.com/",
  "verify_access": true,
  "warn_unknown_mime_types": false
}
```


Summary of API Endpoints

/api/collections/collection/datasources : [list](#) or [create](#) data sources in a particular collection

/api/collections/collection/datasources/id : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

 GET /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters


None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

 POST /api/collections/collection/datasources

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content

None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Windows Shares Data Sources

The Windows Shares data source allows crawling SMB or CIFS filesystems.

As of v2.6.2, this data source supports incremental crawling, which means that it will keep record of the URIs for each document it accesses. In subsequent crawl attempts, the URIs found will be compared with the URIs on record and documents will be added, updated or removed accordingly.

Additionally, certain types of crawl failures may be counted twice in crawl statistics. If a document was not accessible because access was denied to the crawler, it will be counted once in the counter for Access Denied, and a second time as Failure since the content could not be retrieved.

Information about how LucidWorks Search interprets Access Control Lists for a Windows Share, is available at [Crawling Windows Shares with Access Control Lists](#).

- [Windows Shares \(SMB\) Data Source Attributes](#)
 - [Common Data Source Attributes](#)
 - [Windows Shares \(SMB\) Type-Specific Attributes](#)
- [Summary of API Endpoints](#)
 - [Get a List of Data Sources](#)
 - [Create a Data Source](#)
 - [Get Data Source Details](#)
 - [Update a Data Source](#)
 - [Delete a Data Source](#)

Windows Shares (SMB) Data Source Attributes

Common Data Source Attributes

Because all data sources share the same framework, there are a number of shared attributes between each. These should be combined with the type-specific attributes below when creating or updating an S3 data source.

These attributes are used for all data source types (except where specifically noted).

Expand the table of attributes common to most data sources...

General Attributes

The general attributes define the data source name, type, crawler to be used, and collection, among other details.

Expand the table of general attributes ...

Key	Type	Required	Default	Description
id	32-bit integer	No	Auto-assigned	The numeric ID for this data source.

Key	Type	Required	Default	Description
type	string	Yes	Null	<p>The type of this data source. Valid types are:</p> <ul style="list-style-type: none"> • file for a filesystem (remote or local, but must be paired with the correct crawler, as below) • web for HTTP or HTTPS web sites • jdbc for a JDBC database • solrxml for files in Solr XML format • sharepoint for a SharePoint repository • smb for a Windows file share (CIFS) • hdfs for a Hadoop filesystem • s3 for a native S3 filesystem • s3h for a Hadoop-over-S3 filesystem • azure_blob for an Azure Blob • azure_table for an Azure Table • mongodb for a MongoDB instance • push for an externally-managed data source • twitter_stream for a Twitter stream • hadoop for high-volume crawling of a Hadoop filesystem. Note that this type is used with several crawlers, which are customized for each distribution of Hadoop that LucidWorks Search supports.
crawler	string	Yes	Null	<p>Crawler implementation that handles this type of data source. The crawler must be able to support the specified <code>type</code>. Supported types for each crawler is indicated in <i>italics</i> in the list below. Valid crawlers are:</p> <ul style="list-style-type: none"> • lucid.aperture for <i>web</i> and <i>file</i> types • lucid.fs for <i>file</i>, <i>smb</i>, <i>hdfs</i>, <i>s3h</i>, <i>s3</i>, and <i>ftp</i> types • lucid.gcm for <i>sharepoint</i> type • lucid.jdbc for <i>jdbc</i> type • lucid.solrxml for <i>solrxml</i> type • lucid.azureblob for <i>azure_blob</i> type • lucid.azuretable for <i>azure_table</i> type • lucid.mongodb for <i>mongodb</i> type • lucid.push for <i>push</i> type • lucid.twitter.stream for <i>twitter_stream</i> type

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • lucid.twitter.search for <i>twitter_search</i> type • lucid.hadoop.apache1 for <i>hadoop</i> type with Apache Hadoop v1.x • lucid.hadoop.apache2 for <i>hadoop</i> type with Apache Hadoop v2.x • lucid.hadoop.cloudera for <i>hadoop</i> type with Cloudera CDH • lucid.hadoop.intel for <i>hadoop</i> type with Intel Distribution for Hadoop • lucid.hadoop.mapr for <i>hadoop</i> type with MapR Hadoop • lucid.hadoop.pivotal for <i>hadoop</i> type with Pivotal Hadoop
collection	string	Yes	Null	The name of the document collection that documents will be indexed into.
name	string	Yes	Null	A human-readable name for this data source. Names may consist of any combination of letters, digits, spaces and other characters. Names are case-insensitive, and do not need to be unique: several data sources can share the same name.
category	string	No	Null	The category of this data source: Web, FileSystem, Jdbc, SolrXml, SharePoint, External, or Other. For informational purposes only.

Crawler Output

For most search applications, the default crawler output may be sufficient. With the default implementation, the `output_type` is set to "solr", and the `output_args` are the location of Solr, which is interpreted from `master.conf` as the setting of the `LWE-Core` component, and some performance settings that can be modified to improve performance as needed. However, if using LucidWorks Big Data, or integrating with another system that will consume the crawler output, you may want to modify these settings accordingly.

Expand the table of Crawler Output options...

Key	Type	Required	Default	Description
output_type	string	No	"solr"	<p>Advanced. Defines the way crawl output is handled. 7 types are supported:</p> <ul style="list-style-type: none"> • solr: The output will be sent to Solr for indexin

Key	Type	Required	Default	Description
				<ul style="list-style-type: none"> • NULL: The crawl output will be discarded. This as all upper-case. • com.lucid.crawl.impl.FileUpdateController: be sent to a file. • com.lucid.crawl.script.ScriptPreprocessor: This will allow a script to be run on the content to Solr for indexing. Only Javascript is supported. The script name is provided in <code>output_args</code>. • com.lucid.sda.hbase.lws.HBaseUpdateController: output will be sent to an HBase implementation <i>in conjunction with LucidWorks Big Data only</i>. <p>Alternatively, it could be another a fully-qualified custom implementation of <code>UpdateController</code>, created with a custom connector.</p>
output_args	string	No	See description	<p>Advanced. Defines where crawler output should be sent. It is dependent on the <code>output_type</code> selected.</p> <ul style="list-style-type: none"> • <code>output_type</code> is "solr": A few parameters are pre-defined, <code>output_args</code> will default to the Solr instance defined in <code>master.conf</code> and the collection that uses the example, if LucidWorks has been installed in <code>/opt/lucidworks</code> and creating the data source for collection1, the URL would be http://127.0.0.1:8888/solr/collection1). Two additional parameters are possible: <ul style="list-style-type: none"> • "threads": Defines the number of concurrent threads to use for sending updates. This does not decrease performance while crawling a data source, but throughput when updating Solr via SolrJ. The default is 2. • "buffer": Defines the number of documents to buffer before sending to SolrJ in bulk, which can reduce the number of calls to Solr. The default is 1, which means no additional buffering. Increasing this value to higher than one has little impact on performance when the number of threads is 1; the performance benefits are usually seen when <code>threads=1</code>, but at the cost of increased JVM memory consumption. • When using "threads" and "buffer" in combination, express them as key=value pairs, separated by a space with no whitespace between them. For example, <code>threads=2 buffer=1</code>.

Key	Type	Required	Default	Description
				<p>"output_args": "buffer=2, threads=10" (and will use the default Solr location). If the values is missing, the default is used.</p> <ul style="list-style-type: none"> output_type is "com.lucid.crawl.impl.FileUpdate": The output_args must be a URI string for a file point to either a directory (which must exist) or will be created during the crawl (which must not exist the crawl). The path will be interpreted as either absolute paths should be used whenever possible. Relative paths interpreted relative to the working directory of the component, which is \$LWS_HOME. output_type is "com.lucid.crawl.script.ScriptPreprocessor": The output_args are a script name, in the form "name", without any file extension. The system will look for the script name with a .js file extension. Only Javascripts are allowed at this time. The location of the script must be \$LWS_HOME/conf/data source output_type is "com.lucid.sda.hbase.lws.HBaseUpdateConnector": output_args must be the host:port of the ZooKeeper again <i>used with LucidWorks Big Data only</i>. A value is interpreted by HBase, and will be similar to localhost <p>If using a custom implementation of UpdateConnector attribute can be however you defined its use in</p>

Field Mapping

The output also includes the field mapping for the data source, which is modifiable as part of the regular data source [update API](#). The mappings for a data source can also be updated with the [Field Mapping](#) API. Note that not all data sources support field mapping.

Expand the table of Field Mapping options...

The data source attribute `mapping` contains a JSON map with the following keys and values:

Key	Type	Required	Default	Description
mapping	JSON string-string	No	See list of attributes in this table	A map where keys are case-insensitive names of the original metadata key names, and values are case-sensitive names of fields that make sense in the

Key	Type	Required	Default	Description
				current schema. These target field names are verified against the current schema and if they are not valid these mappings are removed. Please note that null target names are allowed, which means that source fields with such mappings will be discarded.
datasource_field	string	No	"data_source"	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	No	"null"	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	No	"attr"	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore).
literals		No	Null	

Key	Type	Required	Default	Description
	JSON string-string			An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	No	"true"	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	No	See description.	<p>The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. See the list in the section on Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p>

Key	Type	Required	Default	Description
				<ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	No	"acl": true, "author": true, "body": false, "dateCreated": false, "description": false, "fileSize": false, "mimeType": false, "title": false	<p>A map of target field names that is automatically initialized from the schema based on the target field's multiValued attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them</p>

Key	Type	Required	Default	Description
				<p>along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as multiValued=false then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, </div>

Key	Type	Required	Default	Description
				<p>so that the resulting field has only single concatenated value.</p> <ul style="list-style-type: none"> For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	No	"false"	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and</p>

Key	Type	Required	Default	Description
				SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, Push, Twitter Search or Twitter Stream data source types.
types	JSON string-string	No	"date": "DATE", "datecreated": "DATE", "filesize": "LONG", "lastmodified": "DATE"	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	No	"id"	Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the

Key	Type	Required	Default	Description
				<p><code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., batch processing is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With push data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	No	"true"	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents</p>

Key	Type	Required	Default	Description
				are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).

Optional Commit Rules

Expand the table of commit options...

The following attributes are optional and relate to when new documents will be added to the index:

Required	Default	Key	Type	Description
commit_within	integer	No	900000	Number of milliseconds that defines the maximum interval between commits while indexing documents. The default is 900,000 milliseconds (15 minutes).
commit_on_finish	boolean	No	True	When true (the default), then commit will be invoked at the end of crawl.

Batch Processing

The following attributes control batch processing and are also optional.

Expand for the Batch Processing options...

See also [Processing Documents in Batches](#) as some crawlers only support a subset of batch processing options. Note that the [MapR High Volume Data Sources](#) and [High-Volume HDFS Data Sources](#) do **not support** any kind of batch processing.

Key	Type	Required	Default	Description
parsing	boolean	No	True	When true (the default), the crawlers will parse rich formats immediately. When false , other processing is skipped and raw input documents are stored in a batch.
indexing	boolean	No	True	When true (the default), then parsed documents will be sent immediately for indexing. When false , parsed documents will be stored in a batch.
caching	boolean	No	False	When true , both raw and parsed documents will always be stored in a batch, in addition to any other requested processing. If false (the default), then batch is not created and documents are not preserved unless as a result of setting other options above.

Windows Shares (SMB) Type-Specific Attributes

When creating a data source of type **smb**, the value **lucid.fs** must be supplied for the `crawler` attribute, described in the section on [common attributes](#).

Key	Type	Required	Default
ad_cache_groups	boolean	No	False
ad_connect_timeout	integer	No	3000
ad_context_factory	string	No	com.sun.jndi.ldap.LdapCtxFactory
ad_credentials	string	No	Null
ad_group_base_dn	string	No	Null
ad_group_filter	string	No	(&(objectclass=group))
ad_read_timeout	integer	No	5000
ad_read_token_groups	boolean	No	true

Key	Type	Required	Default
ad_referral	string	No	follow
ad_security_authentication	string	No	simple
ad_url	string	No	Null

Key	Type	Required	Default
ad_user_base_dn	string	No	Null
ad_user_filter	string	No	(&(objectclass=user)(userPrincipalName=
ad_user_principal_name	string	No	Null
add_failed_docs	boolean	No	False
bounds	string	No	None
cache_element_expiration_time	integer	No	7200
crawl_depth	32-bit integer	No	-1

Key	Type	Required	Default
crawl_item_timeout	integer	No	600000
enable_security_trimming	boolean	No	False
enable_SIDs_cache	boolean	No	True
exclude_paths	list of strings	No	Empty
include_extensions	list of strings	No	Empty
include_paths	list of strings	No	Empty

Key	Type	Required	Default
index_directories	boolean	No	False
max_bytes	long	No	10,485,760
max_docs	integer	No	-1
max_threads	integer	No	1
maximum_connections	integer	No	1000
password	string	Yes	Null

Key	Type	Required	Default
persisted_URIs_transaction_size	integer	No	1000
url	string	Yes	Null
username	string	Yes	Null
verify_access	boolean	No	True
windows_domain	string	No	Null



The `include_paths` and `exclude_paths` attributes for all Remote File System data sources use [Java Regular Expressions](#).

There are a number of SMB related settings that can be controlled by using Java System Properties. See <http://jcifs.samba.org/src/docs/api/overview-summary.html#scp> for more information about those settings. Currently the easiest way to specify these System Properties is to edit the `$LWS_HOME/conf/master.conf` file and modify the value for the property `lwecore.jvm.params`. A [restart](#) of LucidWorks Search is needed to make the changes visible.

Example SMB data source (without mapping attributes):

```
{
  "ad_cache_groups": false,
  "ad_connect_timeout": 3000,
  "ad_context_factory": "com.sun.jndi.ldap.LdapCtxFactory",
  "ad_credentials": null,
  "ad_group_base_dn": null,
  "ad_group_filter": "(&(objectclass=group))",
  "ad_read_timeout": 5000,
  "ad_read_token_groups": true,
  "ad_referral": "follow",
  "ad_security_authentication": "simple",
  "ad_url": null,
  "ad_user_base_dn": null,
  "ad_user_filter": "(&(objectclass=user)(sAMAccountName={0}))",
  "ad_user_principal_name": null,
  "add_failed_docs": false,
  "bounds": "tree",
  "cache_element_expiration_time": 7200,
  "caching": false,
  "category": "FileSystem",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawl_item_timeout": 600000,
  "crawler": "lucid.fs",
  "enable_SIDs_cache": true,
  "enable_security_trimming": false,
  "exclude_paths": [],
  "id": "35e05a192b014659800192972bd85129",
  "include_extensions": [],
  "include_paths": [],
  "index_directories": false,
  "indexing": true,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_cache_size": 1000,
  "max_docs": -1,
  "max_threads": 1,
```

```
{
  "maximum_connections": 1000,
  "name": "Windows Share",
  "output_args": "threads=2,buffer=1",
  "output_type": "solr",
  "parsing": true,
  "password": "pass",
  "persisted_URIs_transaction_size": 1000,
  "type": "smb",
  "url": "smb://mnt/share/",
  "username": "sally",
  "verify_access": true,
  "windows_domain": null
}
```

Summary of API Endpoints

`/api/collections/collection/datasources` : [list](#) or [create](#) data sources in a particular collection

`/api/collections/collection/datasources/id` : [update](#), [remove](#), or [get details](#) for a particular data source

This summary shows the API calls available, but does not provide examples. To see example calls using this API, please review the [Data Sources](#) page.

Get a List of Data Sources

GET `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

A JSON map of attributes, depending on data source type (see above).

Create a Data Source

POST `/api/collections/collection/datasources`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input content

JSON block with all required attributes. The ID field, if defined, will be ignored.


Output

Output Content

JSON representation of new data source.

Get Data Source Details

 GET /api/collections/collection/datasources/ id

 This call requires knowing the ID of the data source. There is no way to query for the ID by using the name, so the only way to find the id of a data source is use the API call to [get a list of data sources](#).

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None.

Input content


None

Output

Output Content

A JSON map of all data source attributes.

Update a Data Source

 PUT /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Input content


JSON block with either all attributes or just those that need updating. The attributes `type` (data source type), `crawler` (crawler type), and `id` (data source ID) cannot be updated.

Output


Output Content

None

Delete a Data Source

 The Data Source DELETE command will delete documents associated with the data source as of v2.5 (in prior versions it did not). To keep the documents, add `keep_docs=true` to the delete request, after the id. For example:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/4?keep_docs=true
```

 DELETE /api/collections/collection/datasources/ id

Input

Path Parameters

Key	Description
collection	the collection name.
id	The data source ID.

Query Parameters

None

Input content

None

Output

Output Content

None

Field Mapping

Field mapping settings can be managed with the Data Sources API, but there is also a way to manipulate mappings and related settings with a dedicated API for field mapping.

Mapping allows control over how documents are indexed on a per-data source basis. If there are fields in the incoming documents that should be indexed to a specific field in the schema, field mapping provides a way to make that happen.

Other changes can be made during the mapping process. For example, you can define a specific dynamic field rule to be used when there isn't a matching field in the schema. You can define a default field to map incoming content to if there is no other explicit mapping. See the details below for more examples of what can be done with field mapping in data sources.

- [API Entry Points](#)
- [List Field Mapping Settings](#)
- [Update Field Mapping Settings](#)
- [List a Specific Mapping Part](#)
- [List a Key for a Mapping Part](#)
- [Remove a Key for a Mapping Part](#)
- [Remove a Mapping Part](#)
- [Remove All Mappings](#)

API Entry Points

`/api/collections/collection/datasources/id/mapping:` [list](#), [update](#) or [remove](#) mapping settings

`/api/collections/collection/datasources/id/mapping/part:` [list](#) or [remove](#) a specific part of the mapping settings

`/api/collections/collection/datasources/id/mapping/part/key:` [list](#) or [remove](#) a specific mapping setting, within a *part*

List Field Mapping Settings

 GET `/api/collections/collection/datasources/id/mapping`

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.

Query Parameters

None.

Output**Output Content**

A JSON List of Maps mapping keys to values.

Key	Type	Description
datasource_field	string	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore). For example, if the <code>dynamic_field</code> is defined as "attr_" and the field name is "updated_price", the resulting field would be "attr_updated_price".
literals	JSON string-string	An optional map that can specify static pairs of keys and values to be added to output documents. If defined, these are applied as the last step of the overall field mapping.
lucidworks_fields	boolean	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	The mappings section contains a list of source fields and the target fields they will be mapped to. Several mappings are defined by default. For details, see the list in the section on

Key	Type	Description
		<p>Field Mapping in the Overview of Crawling.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p> <p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	<p>A map of target field names that is automatically initialized from the schema based on the target field's <code>multiValued</code> attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as <code>multiValued=false</code> then only the longest (probably </div>

Key	Type	Description
		<p>most specific) value is retained, and all other values are discarded.</p> <ul style="list-style-type: none"> • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If multiValued=false in the target schema, then only the first remaining value will be retained, and all other values are discarded. • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the <code>lucid.fs</code>, <code>lucid.aperture</code>, and <code>lucid.gcm</code> crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, External or Twitter data source types.</p>
types	JSON string-string	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are</p>

Key	Type	Description
		<p>assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p> <ul style="list-style-type: none">• Any class with <code>DateField</code> becomes <code>DATE*</code> Any class that ends with <code>*DoubleField</code> becomes <code>DOUBLE</code>• Any class that ends with <code>*FloatField</code> becomes <code>FLOAT</code>• Any class that ends with <code>*IntField</code> or <code>*ShortField</code> becomes <code>INT</code>• Any class that ends with <code>*LongField</code> becomes <code>LONG</code>• Anything else not listed above becomes <code>STRING</code>
unique_key	string	<p>Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With external data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).</p>

Examples

Input

```
curl
http://localhost:8888/api/collections/collection1/datasources/e4228671b8d446afb9d8a23124a3
```


Output

```
{
  "datasource_field": "data_source",
  "default_field": null,
  "dynamic_field": "attr",
  "literals": {},
  "lucidworks_fields": true,
  "mappings": {
    "acl": "acl",
    "author": "author",
    "batch_id": "batch_id",
    "body": "body",
    "content-encoding": "characterSet",
    "content-length": "fileSize",
    "content-type": "mimeType",
    "contentcreated": "dateCreated",
    "contentlastmodified": "lastModified",
    "contributor": "author",
    "crawl_uri": "crawl_uri",
    "created": "dateCreated",
    "creator": "creator",
    "date": null,
    "description": "description",
    "filelastmodified": "lastModified",
    "filename": "fileName",
    "filesize": "fileSize",
    "fullname": "author",
    "fulltext": "body",
    "keyword": "keywords",
    "last-modified": "lastModified",
    "last-printed": null,
    "lastmodified": "lastModified",
    "lastmodifiedby": "author",
    "links": null,
    "messagesubject": "title",
    "mimetype": "mimeType",
    "name": "title",
    "page-count": "pageCount",
    "pagecount": "pageCount",
    "plaintextcontent": "body",
    "plaintextmessagecontent": "body",
    "slide-count": "pageCount",
```

```
    "slides": "pageCount",
    "subject": "subject",
    "title": "title",
    "type": null,
    "url": "url"
  },
  "multi_val": {
    "acl": true,
    "author": true,
    "body": false,
    "dateCreated": false,
    "description": false,
    "fileSize": false,
    "mimeType": false,
    "title": false
  },
  "original_content": false,
  "types": {
    "date": "DATE",
    "datecreated": "DATE",
    "filesize": "LONG",
    "lastmodified": "DATE"
  },
  "unique_key": "id",
  "verify_schema": true
}
```

[Back to Top](#)

Update Field Mapping Settings

 PUT /api/collections/collection/datasources/id/mapping

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.

Query Parameters

None.

Input Parameters

Key	Type	Description
datasource_field	string	A prefix for index fields that are needed for LucidWorks faceting and data source management. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.
default_field	string	The field name to use if source name doesn't match any mapping. If null, then <code>dynamicField</code> will be used, and if that is null too then the original name will be returned.
dynamic_field	string	If not null then source names without specific mappings will be mapped to <code>dynamicField_sourceName</code> , after some cleanup of the source name (non-letter characters are replaced with underscore). For example, if the <code>dynamic_field</code> is defined as "attr_" and the field name is "updated_price", the resulting field would be "attr_updated_price".
literals	JSON string-string	An optional map that can specify static pairs of keys and values to be added to output documents.
lucidworks_fields	boolean	If true , the default, then the field mapping process will automatically add LucidWorks-specific fields (such as <code>data_source</code> and <code>data_source_type</code>) to the documents. There may be some cases where the data source information is already added to the documents, such as with Solr XML documents, where this setting should be false . However, without this information, LucidWorks will not be able to properly identify documents from a specific data source and would not be able to show accurate document counts, display the documents in facets, or delete documents if necessary.
mappings	JSON string-string	<p>The mappings section contains a list of source fields and the target fields they will be mapped to.</p> <p>When the mapping is created or updated, LucidWorks checks the mappings against the <code>schema.xml</code> for the collection and verifies that the target fields exist in the schema.</p>

Key	Type	Description
		<p>During indexing, the field mapping process performs the following steps:</p> <ol style="list-style-type: none"> 1. The mappings are checked for the existence of the source field name. If it exists, it will be mapped to the target field. 2. If the source field name does not exist in the mappings, the <code>schema.xml</code> for the collection is checked. If the source field name exists in the schema, it will be indexed to that field. 3. If a <code>dynamic_field</code> has been defined, a dynamic field will be created according to the dynamic field rule. 4. If a <code>default_field</code> has been defined, the source field will be mapped to the defined default field. 5. If none of these steps has produced a match, the field will be discarded.
multi_val	JSON string-boolean	<p>A map of target field names that is automatically initialized from the schema based on the target field's <code>multiValued</code> attribute. In general, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default.</p> <div> <p>Field mapping normalization is a step applied after all target names for field values have been resolved, including substitution with dynamic or default field names. This step checks that values are compatible with the index schema. The following checks are performed:</p> <ul style="list-style-type: none"> • For the "mimeType" field, : if it is defined as <code>multiValued=false</code> then only the longest (probably most specific) value is retained, and all other values are discarded. • If field type is set to DATE in the field mapping, first the values are checked for validity and invalid values are discarded. If <code>multiValued=false</code> in the </div>

Key	Type	Description
		<p>target schema, then only the first remaining value will be retained, and all other values are discarded.</p> <ul style="list-style-type: none"> • If field type is STRING, and multiValued=false in the target schema, then all values are concatenated using a single space character, so that the resulting field has only single concatenated value. • For all other field types, if multiValued=false and multiple values are encountered, only the first value is retained and all other values are discarded.
original_content	boolean	<p>If true, adds the ability to store the original raw bytes of any document. By default it is false. If this is enabled, a field called "original_content" will be added to each document, and will contain the raw bytes of the original document. The field is subject to normal field mapping rules, which means that if this field is not defined in the <code>schema.xml</code> file, it will be added dynamically as <code>attr_original_content</code> according to the default rules of field mapping. If the "attr_" dynamic rule has been removed, this field may be deleted during field mapping if it is not defined in <code>schema.xml</code> (which it is not by default, so possibly should be added, depending on your configuration).</p> <p>The data source types that use the lucid.fs, lucid.aperture, and lucid.gcm crawlers (so, data source types Web, File, SMB, HDFS, S3, S3H, FTP, and SharePoint) are the only ones that support this attribute. It is not possible to store original binary content for the Solr XML, JDBC, External or Twitter data source types.</p>
types	JSON string-string	<p>A map pre-initialized from the current schema. Additional validation can be performed on fields with declared non-string types. Currently supported types are DATE, INT, LONG, DOUBLE, FLOAT and STRING. If not specified fields are assumed to have the type STRING.</p> <p>The map is pre-initialized from the types definition in <code>schema.xml</code> in the following ways:</p>

Key	Type	Description
		<ul style="list-style-type: none"> Any class with DateField becomes DATE* Any class that ends with *DoubleField becomes DOUBLE Any class that ends with *FloatField becomes FLOAT Any class that ends with *IntField or *ShortField becomes INT Any class that ends with *LongField becomes LONG Anything else not listed above becomes STRING
unique_key	string	<p>Defines the document field to use as the unique key in the Solr schema. For example, if the schema uses "id" as the unique key field name, and the <code>unique_key</code> attribute is set to "url", then field mapping will map "url" to "id". By default, this will be adjusted to match the <code>schema.xml</code> value. However, in cases where no indexing will be performed (i.e., a batch crawl is being performed), the <code>schema.xml</code> is not available for checking so it may be useful to edit this parameter manually to fit the expected schema value. If performing a normal crawl (i.e., the crawler finds the documents, parses them, and passes them along for indexing), this field should be left as the default. With external data sources, this parameter would map a field from the incoming documents to be the unique key for all documents.</p>
verify_schema	boolean	<p>If true, the default, then field mapping will be validated against the current schema at the moment the crawl job is started. This may result in dropping some fields or changing their multiplicity so they conform to the current schema. The modified mapping is not propagated back to the data source definition (i.e., it is not saved permanently). In this way, the schema can be modified without having to modify the data source mapping definition; however, it may also be more difficult to learn what the final field mapping was. If this value is false, then the field mapping rules are not verified and are applied as is, which may result in exceptions if documents are added that don't match the current schema (e.g., incoming documents have multiple values in a field when the schema expects a single value).</p>



When updating any mapping setting, all previously defined settings for the entire field mapping configuration must be sent with the updated attribute or pre-defined settings, including system defaults, will be removed.

Output

Output Content

None.

Examples


Input

Change the mapping of the incoming field "type" from "null" to the schema field "mimeType":

```
curl -X PUT -H 'Content-type: application/json' -d '{"dynamic_field":"attr",
"datasource_field":"data_source", "literals":{}, "multi_val":{"fileSize":false,
"body":false, "author":true, "title":false, "acl":true, "description":false,
"dateCreated":false, "mimeType":false}, "verify_schema":true, "default_field":null,
"mappings":{"slide-count":"pageCount", "content-type":"mimeType", "body":"body",
"slides":"pageCount", "subject":"subject", "plaintextmessagecontent":"body",
"lastmodified":"lastModified", "lastmodifiedby":"author",
"content-encoding":"characterSet", "type":"mimeType", "date":null, "creator":"creator",
"author":"author", "title":"title", "mimetype":"mimeType", "created":"dateCreated",
"plaintextcontent":"body", "pagecount":"pageCount", "contentcreated":"dateCreated",
"description":"description", "contributor":"author", "name":"title",
"filelastmodified":"lastModified", "fullname":"author", "fulltext":"body",
"messagesubject":"title", "last-modified":"lastModified", "acl":"acl",
"keyword":"keywords", "contentlastmodified":"lastModified", "last-printed":null,
"links":null, "url":"url", "batch_id":"batch_id", "crawl_uri":"crawl_uri",
"filesize":"fileSize", "page-count":"pageCount", "content-length":"fileSize",
"filename":"fileName"}, "unique_key":"id", "types":{"filesize":"LONG",
"lastmodified":"DATE", "datecreated":"DATE", "date":"DATE"}, "original_content":false,
"lucidworks_fields":true}'
http://localhost:8888/api/collections/collection1/datasources/82b8c0e1d9114eceedf592986f66
```

[Back to Top](#)

List a Specific Mapping Part

 GET /api/collections/collection/datasources/id/mapping/part

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.
part	

Key	Description
	The section of the mapping definition to list. The parts are any of the attributes listed in the table in the section List Field Mapping Settings , such as <code>mappings</code> , <code>unique_key</code> and <code>multi_val</code> .

Query Parameters

None.

Output

Output Content

A JSON List of keys to values, which will vary depending on the part. See the section [List Field Mapping Settings](#) for explanations of the parts of the mapping definitions.

Examples

Input

List the settings for the `multi_val` part:

```
curl
http://localhost:8888/api/collections/collection1/datasources/e4228671b8d446afb9d8a23124a3
```

Output

```
{
  "acl": true,
  "author": true,
  "body": false,
  "dateCreated": false,
  "description": false,
  "fileSize": false,
  "mimeType": false,
  "title": false
}
```

[Back to Top](#)

List a Key for a Mapping Part

GET `api/collections/collection/datasources/id/mapping/part/key`

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.
part	The section of the mapping definition to list. The parts are any of the attributes listed in the table in the section List Field Mapping Settings , such as <code>mappings</code> , <code>unique_key</code> and <code>multi_val</code> .
key	The specific subset of the part to list, if the part is a map (with multiple sub-values, such as <code>literals</code> , <code>mappings</code> , <code>multi_val</code> and <code>types</code>) and not a primitive (with a single value, such as <code>default_field</code> or <code>unique_key</code>). Primitives can be returned with the syntax to return a part.

Query Parameters

None.

Output

Output Content

The value for the specified key. This may be a string, integer or boolean depending on the value of the specified key.

Examples

Input

Get the field that 'created' will be mapped to:

```
curl
http://localhost:8888/api/collections/collection1/datasources/e4228671b8d446afb9d8a23124a3
```

Output

```
dateCreated
```

[Back to Top](#)

Remove a Key for a Mapping Part

 DELETE /api/collections/collection/datasources/id/part/key

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.
part	The section of the mapping definition to list. The parts are any of the attributes listed in the table in the section List Field Mapping Settings , such as <code>mappings</code> , <code>unique_key</code> and <code>multi_val</code> .
key	The specific subset of the part to list, if the part is a map (with multiple sub-values, such as <code>literals</code> , <code>mappings</code> , <code>multi_val</code> and <code>types</code>) and not a primitive (with a single value, such as <code>default_field</code> or <code>unique_key</code>). Primitives can be removed with the syntax to return a part.

Query Parameters

None.

Output

Output Content

None.

Examples


Input

Remove the mapping for the 'last-modified' field:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/127d4e96a8f649d4a90e962e94c3
```

[Back to Top](#)

Remove a Mapping Part

 `DELETE /api/collections/collection/datasources/id/part`

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.
part	

Key	Description
	The section of the mapping definition to list. The parts are any of the attributes listed in the table in the section List Field Mapping Settings , such as <code>mappings</code> , <code>unique_key</code> and <code>multi_val</code> .

Query Parameters

None.

Output

Output Content

None.

Examples

Input


Remove the settings in the `types` part of the mapping:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/127d4e96a8f649d4a90e962e94c3
```

[Back to Top](#)

Remove All Mappings

This will reset all of the mapping settings to their default values.

 `DELETE /api/collections/collection/datasources/id/mapping`

Input

Path Parameters

Key	Description
collection	The name of the collection.
id	The ID of the data source.

Query Parameters

None.

Output

Output Content

None.

Examples**Input**

```
curl -X DELETE  
http://localhost:8888/api/collections/collection1/datasources/127d4e96a8f649d4a90e962e94c3
```

[Back to Top](#)

Data Source Schedules

Each data source has a corresponding schedule that determines when that particular data source will be updated. Typically, schedules are recurring, so LucidWorks can index new documents from a data source. However, schedules can also be programmed for a single point in time.

A schedule is a property of a data source. As such, it does not have an individual ID. Instead, the API references the ID of the data source itself. The schedule is created at the same time as the data source creation, so there is no mechanism to create a data source schedule.



A schedule set with this API may display in the [Admin UI](#) as "custom" if the options selected do not match those available via the UI. Some examples of cases where this might happen:

- The period is set to something other than hourly (3600 seconds), daily (86,400 seconds) or weekly (604,800 seconds).
- The start_time is set to a day that's not today.
- The start_time is set to an hour that is not this hour.

If the schedule is shown as "custom" in the UI, it is still possible to edit it via the UI. However the options will be limited to the options available through the UI at the current time.

Schedules can be automatically disabled by LucidWorks Search if they fail on a consistent basis, but not all crawl failures will trigger a deactivation of the data source schedule. If there is a missing parameter or other fatal error in the configuration of the data source that will always cause an error, that will cause the deactivation of the schedule. Similarly, if there is another circumstance that causes the crawl to fail when it launches (such as, the required crawler is no longer installed with LucidWorks, or the Solr handler is missing), and the scheduled task fails three times, the schedule will be deactivated. If a crawl consistently runs for a time and fails at some point, that will not trigger deactivation of the schedule.

- [API Entry Points](#)
- [Get a Data Source Schedule](#)
- [Update a Data Source Schedule](#)

API Entry Points

/api/collections/collection/datasources/id/schedule: [Get](#) or [update](#) this data source's schedule.

Get a Data Source Schedule

 GET /api/collection/collection/datasources/id/schedule

Input

Path Parameters

Key	Description
collection	The collection name
id	The data source ID

Query Parameters

None

Output

Output Content

Key	Type	Description
start_time	date string	The start date for this schedule, in the format <code>yyyy-MM-dd'T'HH:mm:ss' +/- 'hhmm</code> . The <code>' +/- '</code> is adjusted for the time zone relative to UTC.
period	64-bit integer	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	Currently always <code>crawl</code> .
active	boolean	If true , this schedule will be run at the next scheduled time.

Examples

Find out when data source 8 will be indexed next.

Input

```
curl http://localhost:8888/api/collections/collection1/datasources/8/schedule
```

Output

The data source is scheduled to run every Monday at midnight:

```
{
  "active": true,
  "period": 604800,
  "start_time": "2012-12-17T00:00:00+0000",
  "type": "crawl"
}
```

[Back to Top](#)

Update a Data Source Schedule

PUT /api/collections/collection/datasources/id/schedule

Input

Path Parameters

Key	Description
collection	The collection name
id	The data source ID

Query Parameters

None

Input Content

Key	Type	Required	Default	Description
start_time	date string	Yes	create date/time of the data source	The start date for this schedule, in the format yyyy-MM-dd 'T' HH:mm:ss '+/-' hhmm. The API can accept a relative time of "now". The '+/-' is for the time zone relative to UTC, and also entered without quotes.
period	64-bit integer	No	0	The number of seconds in between repeated invocations of this schedule; set to 0 if this should only occur once.
type	string	Yes	crawl	Currently always <code>crawl</code> (required).
active	boolean	No	false	If true , this schedule will be run at the next scheduled time.

Output

Output Content

None

Examples

Input

```
curl -X PUT -H 'Content-type: application/json' -d '{"period": 300,"type": "crawl","start_time": "2011-03-18T12:10:32-0700","active": true}' http://localhost:8888/api/collections/collection1/datasources/8/schedule
```

Output

None.

Data Source Jobs

The Data Sources Jobs API allows direct control over the life cycle of data source crawl jobs.


- [API Entry Points](#)
- [Get the Status of a Data Source in a Collection](#)
- [Start Crawling a Data Source in a Collection](#)
- [Stop Crawling a Data Source in a Collection](#)
- [Get the Status of All Data Sources in a Collection](#)
- [Start Crawling All Data Sources in a Collection](#)
- [Stop Crawling All Data Sources in a Collection](#)

API Entry Points

`/api/collections/collection/datasources/id/job`: [get the status](#) of, [start](#), or [stop](#) crawling a data source for a particular collection

`/api/collections/collection/datasources/all/job`: [get the status](#) of, [start](#), or [stop](#) crawling all data sources for a particular collection

Get the Status of a Data Source in a Collection

 GET `/api/collections/collection/datasources/id/job`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Output

Output Content

Key	Description
batch_job	If false , the content crawled will be added to the index.
crawl_started	The date and time the crawl started.
crawl_state	The current state of the job. Entries are FINISHED, STOPPED, or RUNNING.

Key	Description
crawl_stopped	The date and time the crawl stopped.
id	The unique id of the datasource.
job_id	The ID of the job itself.
num_access_denied	The number of documents that could not be accessed because of file permissions or wrong authentication.
num_deleted	The number of documents that were removed from the index.
num_failed	The number of documents that could not be parsed.
num_filter_denied	The number of documents that could not be accessed because of inclusion or exclusion rules.
num_new	The number of documents considered "new".
num_not_found	The number of documents the crawler expected to find (because of a link from a known document, from a symlink, or a redirect) but the remote server responded with HTTP 404 NOT_FOUND or "file missing".
num_robots_denied	The number of documents that could not be crawled because of robots.txt rules.
num_total	The total number of documents found during the last crawl.
num_unchanged	The number of documents that were not changed.
num_updated	The number of documents that were updated.

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/datasources/6/job
```

Output

```
{
  "batch_job": false,
  "crawl_started": "2012-02-06T18:40:12+0000",
  "crawl_state": "FINISHED",
  "crawl_stopped": "2012-02-06T18:42:19+0000",
  "id": 6,
  "job_id": "6",
  "num_access_denied": 0,
  "num_deleted": 0,
  "num_failed": 2,
```

```
"num_filter_denied": 0,  
"num_new": 1099,  
"num_not_found": 0,  
"num_robots_denied": 0,  
"num_total": 1101,  
"num_unchanged": 0,  
"num_updated": 0  
}
```

[Back to Top](#)

Start Crawling a Data Source in a Collection

PUT /api/collections/collection/datasources/id/job

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Output

Output Content

None.

Examples

Input

```
curl -X PUT http://localhost:8888/api/collections/collection1/datasources/8/job
```

Output

None.

[Back to Top](#)

Stop Crawling a Data Source in a Collection

DELETE /api/collections/collection/datasources/id/job

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None.

Output

Output Content

None.

Examples

Input


```
curl -X DELETE http://localhost:8888/api/collections/collection1/datasources/8/job
```

Output

None.

[Back to Top](#)

Get the Status of All Data Sources in a Collection

 GET /api/collections/collection/datasources/all/job

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Description
batch_job	If false , the content crawled will be added to the index.
crawl_started	The date and time the crawl started.
crawl_state	The current state of the job. Entries are FINISHED, STOPPED, or RUNNING.
crawl_stopped	The date and time the crawl stopped.
id	The unique id of the datasource.
job_id	The ID of the job itself.
num_access_denied	The number of documents that could not be accessed because of file permissions or wrong authentication.
num_deleted	The number of documents that were removed from the index.
num_failed	The number of documents that could not be parsed.
num_filter_denied	The number of documents that could not be accessed because of inclusion or exclusion rules.
num_new	The number of documents considered "new".
num_not_found	The number of documents the crawler expected to find (because of a link from a known document, from a symlink, or a redirect) but the remote server responded with HTTP 404 NOT_FOUND or "file missing".
num_robots_denied	The number of documents that could not be crawled because of robots.txt rules.
num_total	The total number of documents found during the last crawl.
num_unchanged	The number of documents that were not changed.
num_updated	The number of documents that were updated.

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/datasources/all/job
```


Output

```
[
  {
    "batch_job": false,
```

```
"crawl_started": "2012-02-06T18:53:53+0000",
"crawl_state": "RUNNING",
"crawl_stopped": null,
"id": 7,
"job_id": "7",
"num_access_denied": 0,
"num_deleted": 0,
"num_failed": 0,
"num_filter_denied": 0,
"num_new": 7,
"num_not_found": 0,
"num_robots_denied": 0,
"num_total": 7,
"num_unchanged": 0,
"num_updated": 0
},
{
  "batch_job": false,
  "crawl_started": "2012-02-06T18:40:12+0000",
  "crawl_state": "FINISHED",
  "crawl_stopped": "2012-02-06T18:42:19+0000",
  "id": 6,
  "job_id": "6",
  "num_access_denied": 0,
  "num_deleted": 0,
  "num_failed": 2,
  "num_filter_denied": 0,
  "num_new": 1099,
  "num_not_found": 0,
  "num_robots_denied": 0,
  "num_total": 1101,
  "num_unchanged": 0,
  "num_updated": 0
}
]
```

[Back to Top](#)

Start Crawling All Data Sources in a Collection

 PUT /api/collections/collection/datasources/all/job

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

None.

Examples

Input


```
curl -X PUT http://localhost:8888/api/collections/collection1/datasources/all/job
```

Output

None.

[Back to Top](#)

Stop Crawling All Data Sources in a Collection

 DELETE /api/collections/collection/datasources/all/job

Input

Path Parameters

Key	Description
collection	The collection name

Query Parameters

None.

Output

Output Content

None.

Examples

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/datasources/all/job
```

Output

None.

[Back to Top](#)

Data Source Status

The Data Source Status API provides a means to get information about whether a data source is currently being processed. This outputs the same information as the [Data Source Jobs API](#) but is available as a way to intermittently check the progress of the job.

- [API Entry Points](#)
- [Get the Status of a Data Source](#)

API Entry Points

`/api/collections/collection/datasources/id/status`: Get this data source's status

Get the Status of a Data Source

🚩 GET `/api/collection/collection/datasources/id/status`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None

Output

Output Content

Key	Description
batch_job	If false , the content crawled will be added to the index.
crawl_started	The date and time the crawl started.
crawl_state	The current state of the job. Entries are FINISHED, STOPPED, or RUNNING.
crawl_stopped	The date and time the crawl stopped.
id	The unique id of the datasource.
job_id	The ID of the job itself.
num_access_denied	The number of documents that could not be accessed because of file permissions or wrong authentication.

Key	Description
num_deleted	The number of documents that were removed from the index.
num_failed	The number of documents that could not be parsed.
num_filter_denied	The number of documents that could not be accessed because of inclusion or exclusion rules.
num_new	The number of documents considered "new".
num_not_found	The number of documents the crawler expected to find (because of a link from a known document, from a symlink, or a redirect) but the remote server responded with HTTP 404 NOT_FOUND or "file missing".
num_robots_denied	The number of documents that could not be crawled because of robots.txt rules.
num_total	The total number of documents found during the last crawl.
num_unchanged	The number of documents that were not changed.
num_updated	The number of documents that were updated.

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/datasources/2/status
```

Output

While the data source is being processed:

```
{
  "batch_job": false,
  "crawl_started": "2012-02-06T18:40:12+0000",
  "crawl_state": "RUNNING",
  "crawl_stopped": null,
  "id": 6,
  "job_id": "6",
  "num_access_denied": 0,
  "num_deleted": 0,
  "num_failed": 2,
  "num_filter_denied": 0,
  "num_new": 227,
  "num_not_found": 0,
  "num_robots_denied": 0,
  "num_total": 229,
```

```
"num_unchanged": 0,  
"num_updated": 0  
}
```

After processing is finished, and the data source is idle:

```
{  
  "batch_job": false,  
  "crawl_started": "2012-02-06T18:40:12+0000",  
  "crawl_state": "FINISHED",  
  "crawl_stopped": "2012-02-06T18:42:19+0000",  
  "id": 6,  
  "job_id": "6",  
  "num_access_denied": 0,  
  "num_deleted": 0,  
  "num_failed": 2,  
  "num_filter_denied": 0,  
  "num_new": 1099,  
  "num_not_found": 0,  
  "num_robots_denied": 0,  
  "num_total": 1101,  
  "num_unchanged": 0,  
  "num_updated": 0  
}
```

Data Source History

The Data Source History API returns historical statistics for previous data source runs. History only returns information about *prior* crawls for a data source. Use [Data Source Jobs](#) or [Data Source Status](#) for details on currently running crawls.


Note that some crawlers are "stateless", meaning the crawler can not be aware of documents deleted or modified between crawls. In crawl statistics, this can mean that deleted or updated documents are not counted as such, or that adding the total number of "new" documents in two different crawls does not equal the number of documents in the index.

- [API Entry Points](#)
- [Get Data Source History](#)

API Entry Points

`/api/collections/name/datasources/id/history`: Get statistics for the last 10 runs of the given data source.

Get Data Source History

 GET `/api/collections/collection/datasources/id/history`

Input

Path Parameters

Key	Description
collection	The collection name.
id	The data source ID.

Query Parameters

None

Output

Output Content

Key	Type	Description
id	integer	The ID of the datasource.
crawl_started	date string	When the crawl began.
crawl_stopped		When the crawl finished.

Key	Type	Description
	date string	
crawl_state	string	The current state of the crawl (RUNNING, FINISHED, or STOPPED).
num_unchanged	32-bit integer	The number of documents found that were not modified and did not need to be indexed.
num_deleted	32-bit integer	The number of documents that were removed from the index because they were no longer found in the source.
num_new	32-bit integer	The number of new documents that were found in the source and added to the index.
num_updated	32-bit integer	The number of existing documents that were found in the source and updated in the index because they were modified since the last time they were indexed.
num_failed	32-bit integer	The number of documents from which the crawler failed to extract text.
num_total	32-bit integer	The total number of documents found.
batch_job	boolean	If false , documents found will be indexed after crawling.
job_id	integer	The ID of the job.

Examples

Input

```
curl http://localhost:8888/api/collections/myCollection/datasources/8/history
```

Output

```
[
  {
    "id": 2,
    "crawl_started": "2011-03-17T22:16:46+0000",
    "num_unchanged": 0,
    "crawl_state": "FINISHED",
    "crawl_stopped": "2011-03-17T22:16:51+0000",
    "job_id": "2",
    "num_updated": 0,
    "num_new": 6,
    "num_failed": 0,
    "num_deleted": 0,
```

```
    "num_total":6,  
    "batch_job":false,  
    "job_id":3  
  },  
  {  
    "id": 2,  
    "crawl_started": "2011-03-18T03: 25:04+0000",  
    "num_unchanged": 0,  
    "crawl_state": "FINISHED",  
    "crawl_stopped": "2011-03-18T 03:25:12+0000",  
    "job_id": "2",  
    "num_updated": 0,  
    "num_new": 6,  
    "num_failed": 0,  
    "num_deleted": 0,  
    "num_total": 6,  
    "batch_job":false,  
    "job_id":2  
  }  
]
```

Data Source Crawl Data Delete

The Data Source Crawl Data API can be used to remove the entire crawl history for a data source. For crawlers, the *crawl history* is the record of which documents have been seen in prior runs. Without a crawl history, when a data source is re-crawled all documents will be treated as "never before seen". This can be useful if field settings have been changed (such as whether the field is stored or not) and a re-crawl of content is required.


This API does not delete the details of prior runs that is shown in the Admin UI or in response to the [Data Source History](#) API (the counts of documents found, updated, deleted, etc.); it only deletes records from the internal crawler database that stores previously seen documents.

- [API Entry Points](#)
- [Delete Crawl History of a Data Source](#)

API Entry Points

`/api/collections/collection/datasources/id/crawldata`: Delete the crawl history (persistent crawl data) for a data source.

Delete Crawl History of a Data Source

 DELETE `/api/collections/collection/datasources/id/crawldata`

Input

Path Parameters

Key	Description
collection	The collection name
id	the data source ID

Query Parameters

None

Output

Output Content

None

Examples

Input

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/3/crawldata
```

Output

None.

Batch Operations

The Batch Operations API allows you to work with crawled batches of content and configure further batch processing jobs. Batches are created by setting the `indexing` attribute of a data source to **false** with the [Data Sources](#) API. For more background information about batch processing, including how to configure a data source for batch crawling, see [Processing Documents in Batches](#).

- [API Entry Points](#)
- [List All Existing Batches](#)
- [Delete All Existing Batches](#)
- [List All Batches For A Specific Crawler Controller](#)
- [Delete All Batches For A Specific Crawler Controller](#)
- [List All Batch Jobs For A Specific Crawler Controller](#)
- [Define A Batch Job For A Specific Crawler Controller](#)
- [Start a Batch Processing Job](#)
- [Get The Status Of A Running Batch Processing Job](#)
- [Stop A Running Batch Processing Job](#)

API Entry Points


`api/collections/collection/batches`: [List](#) or [delete](#) existing batches.

`api/collections/collection/batches/crawler`: [List](#) or [delete](#) all batches managed by a given crawler controller.

`api/collections/collection/batches/crawler/job`: [List](#) all existing or [define](#) a new batch processing job.

`api/collections/collection/batches/crawler/job/batch_id`: [Start](#), [get](#) the status of, or [stop](#) a running batch job.

List All Existing Batches

 GET `api/collections/collection/batches`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Type	Description
num_docs	integer	The number of documents in the batch.
batch_id	string	The batch identifier.
parsed	boolean	Indicates whether the documents in the batch have been parsed.
description	string	The descriptive text you have given the data source.
finish_time	integer	The time at which the batch process was finished.
start_time	integer	The time at which the batch process started.
parsed_docs	integer	The number of parsed documents in the batch.
ds_id	string	The data source identifier.
crawler	string	The crawler controller type for the batch.
collection	string	The name of the collection containing the batch.

Examples

Get a list of all batches:

Input

```
curl http://localhost:8888/api/collections/collection1/batches
```

Output


```
[
{
  "num_docs":0,
  "batch_id":"00000000-0000-0000-0000-057edb4a3fde",
  "parsed":true,
  "description":"CNN",
  "finish_time":0,
  "start_time":1313524585449,
  "parsed_docs":190,
  "ds_id":"39",
  "collection":"collection1",
  "crawler":"lucid.aperture"
},
{

```

```
"num_docs":0,
"batch_id":"00000000-0000-0000-0000-05a6aa4c26e8",
"parsed":false,
"description":"HDFSTestSite",
"finish_time":0,
"start_time":1313524756426,
"parsed_docs":0,
"ds_id":"57",
"collection":"collection1",
"crawler":"lucid.fs"},
{
  "num_docs":0,
  "batch_id":"00000000-0000-0000-0000-057e60bc7733",
  "parsed":false,
  "description":"XMLDS",
  "finish_time":0,
  "start_time":1313524583393,
  "parsed_docs":0,
  "ds_id":"38",
  "collection":"collection1",
  "crawler":"lucid.solrxml"
}
]
```

[Back to Top](#)

Delete All Existing Batches

 DELETE api/collections/collection/batches

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

None.

Examples

Delete all existing batches:

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/batches
```

Output

None.

[Back to Top](#)

List All Batches For A Specific Crawler Controller

🚩 GET api/collections/collection/batches/crawler

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as lucid.aperture.

Query Parameters

None.

Output

Output Content

Key	Type	Description
num_docs	integer	The number of documents in the batch.
batch_id	string	The batch identifier.
parsed	boolean	Indicates whether the documents in the batch have been parsed.
description	string	The descriptive text you have given the data source.
finish_time	integer	The time at which the batch process was finished.
start_time	integer	The time at which the batch process started.
parsed_docs	integer	The number of parsed documents in the batch.
ds_id	string	The data source identifier.
crawler	string	The crawler controller type for the batch.

Key	Type	Description
collection	string	The name of the collection containing the batch.

Examples

List batches for the `lucid.aperture` controller:

Input


```
curl http://localhost:8888/api/collections/collection1/batches/lucid.aperture
```

Output

```
[
{
  "num_docs":0,
  "batch_id":"00000000-0000-0000-0000-012e3da48bde",
  "parsed":true,
  "description":"CNN",
  "finish_time":0,
  "start_time":1313519841161,
  "parsed_docs":120,
  "ds_id":"12",
  "collection":"collection1",
  "crawler":"lucid.aperture"
},
{
  "num_docs":0,
  "batch_id":"00000000-0000-0000-0000-05a4574bf361",
  "parsed":true,
  "description":"FileSystemDS",
  "finish_time":0,
  "start_time":1313524746443,
  "parsed_docs":1068,
  "ds_id":"51",
  "collection":"collection1",
  "crawler":"lucid.aperture"
}
]
```

[Back to Top](#)

Delete All Batches For A Specific Crawler Controller

 DELETE `api/collections/collection/batches/crawler`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .

Query Parameters

None.

Output

Output Content

None.

Examples

Delete all batches for the `lucid.aperture` controller:

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/batches/lucid.aperture
```

Output

None.

[Back to Top](#)

List All Batch Jobs For A Specific Crawler Controller

🚩 GET `api/collections/collection/batches/crawler/job`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .

Query Parameters

None.

Output

Output Content

Key	Type	Description
id	string	The data source identifier
crawl_started	timestamp	The timestamp from the start of the crawling process.
num_total	integer	The number of documents included in the job.
num_unchanged	integer	The number of unchanged documents included in the job.
crawl_state	string	The current state of the crawling process.
crawl_finished	timestamp	The timestamp from the finish of the crawling process.
job_id	string	The job identifier.
batch_job	boolean	Indicates whether the job is a batch job or not.
num_updated	integer	The number of updated documents in the job.
num_new	integer	The number of new documents in the job.
num_failed	integer	The number of documents that the crawler failed to process in the job.
num_deleted	integer	The number of deleted documents in the job.

Examples

List batch jobs for the `lucid.aperture` controller:

Input

```
curl http://localhost:8888/api/collections/collection1/batches/lucid.aperture/job
```

Output

```
[{
  "id": "72",
  "crawl_started": "2011-08-16T20:19:56+0000",
  "num_total": 2,
  "num_unchanged": 0,
  "crawl_state": "FINISHED",
  "crawl_stopped": "2011-08-16T20:19:56+0000",
  "job_id": "00000000-0000-0000-0000-06c639b86e43",
  "batch_job": true,
  "num_updated": 0,
  "num_new": 2,
```

```
"num_failed":0,  
"num_deleted":0  
}]
```

[Back to Top](#)

Define A Batch Job For A Specific Crawler Controller

 PUT `api/collections/collection/batches/crawler/job`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .

Query Parameters

None.

Input Content

Key	Type	Required	Default	Description
batch_id	string	yes	null	The batch identifier, obtained from the batch listing.
collection	string	no	see description	The name of the original collection containing the batch. If this parameter is not included, LucidWorks uses the current collection.
crawler	string	no	see description	The original crawler controller type for the batch. If this parameter is not included, LucidWorks uses the current crawler controller.
ds_id	string	no	null	Defines a data source to use as a template. If you specify a template data source, it will overwrite the field mappings and batch configuration values (parsing, indexing, and caching) for the original data source, as well as the data source identifier. For example, if you are defining a batch job for data source 5, and you set the <code>ds_id</code> value to 4, the new batch job will use the field mappings and batch configuration from data source 4 regardless of

Key	Type	Required	Default	Description
				<p>the configuration you have defined for data source 5. It will also display <code>id:4</code> when you list your batch jobs.</p> <p>If this parameter is not included, LucidWorks uses the original data source field mappings, batch configuration, and identifier.</p>
<code>new_collection</code>	string	no	null	Override for the target collection name, regardless of the batch or data source collection names.
<code>parse</code>	boolean	no	false	If true, LucidWorks parses the batch again, regardless of whether it has already been parsed. If false, the batch is parsed only when <code>index==true</code> and the batch has not been parsed.
<code>index</code>	boolean	no	true	If true, LucidWorks submits the parsed documents for indexing in the target collection. If false, LucidWorks does not index the documents.

Output

Output Content

None.

Examples

Define a batch job for the `lucid.aperture` controller:

Input


```
curl -X PUT -H 'Content-type: application/json' -d '{
  "batch_id": "00000000-0000-0000-1243-4df8a1b27888",
  "collection": "collection1",
  "crawler": "lucid.aperture",
  "ds_id": "4",
  "parse": true,
  "index": true
}' http://localhost:8888/api/collections/collection1/batches/lucid.aperture/job
```

Output

None.

[Back to Top](#)

Start a Batch Processing Job

 PUT `api/collections/collection/batches/crawler/job/batch_id`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .
batch_id	The batch identifier.

Query Parameters

None.

Output

Output Content

None.

Examples

Start a batch processing job:

Input


```
curl -X PUT -H 'Content-type: application/json' -d
'{"batch_id": "00000000-0000-0000-1243-344f2becaaa0", "ds_id": "4", "collection": "collection1"}'
http://localhost:8888/api/collections/collection1/batches/lucid.aperture/job
```

Output

None.

[Back to Top](#)

Get The Status Of A Running Batch Processing Job

 GET `api/collections/collection/batches/crawler/job/batch_id`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .
batch_id	The batch identifier.

Query Parameters

None.

Output

Output Content

Key	Type	Description
id	string	The data source identifier
crawl_started	timestamp	The timestamp from the start of the crawling process.
num_total	integer	The number of documents included in the job.
num_unchanged	integer	The number of unchanged documents included in the job.
crawl_state	string	The current state of the crawling process.
crawl_finished	timestamp	The timestamp from the finish of the crawling process.
job_id	string	The job identifier.
batch_job	boolean	Indicates whether the job is a batch job or not.
num_updated	integer	The number of updated documents in the job.
num_new	integer	The number of new documents in the job.
num_failed	integer	The number of documents that the crawler failed to process in the job.
num_deleted	integer	The number of deleted documents in the job.

Examples

Get the status of a running batch processing job:


Input

--

```
curl
http://localhost:8888/api/collections/collection1/batches/lucid.aperture/job/00000000-0000
```


Output

```
[
  {
    "id": "72",
    "crawl_started": "2011-08-16T20:19:56+0000",
    "num_total": 2,
    "num_unchanged": 0,
    "crawl_state": "FINISHED",
    "crawl_stopped": "2011-08-16T20:19:56+0000",
    "job_id": "00000000-0000-0000-0000-06c639b86e43",
    "batch_job": true,
    "num_updated": 0,
    "num_new": 2,
    "num_failed": 0,
    "num_deleted": 0
  }
]
```

 Running batch jobs also appear in the list of all crawl jobs, but their identifiers correspond to `batch_id` and not to a data source ID.

[Back to Top](#)

Stop A Running Batch Processing Job

 DELETE `api/collections/collection/batches/crawler/job/batch_id`

Input

Path Parameters

Key	Description
collection	The collection name.
crawler	The name of the crawler controller, such as <code>lucid.aperture</code> .
batch_id	The batch identifier.

Query Parameters

None.

Output**Output Content**

None.

Examples

Stop a running batch processing job:

Input

```
curl -X DELETE  
http://localhost:8888/api/collections/collection1/batches/lucid.aperture/job/00000000-0000
```

Output

None.

JDBC Drivers

This functionality is
not available with
LucidWorks Search
on AWS or Azure

LucidWorks Search allows crawling of databases using a JDBC driver compatible with your RDBMS. However, due to licensing constraints, LucidWorks does not ship with a suite of compatible drivers and they must be added to LucidWorks manually. The JDBC Drivers API allows you to upload drivers to LucidWorks Search. Drivers used with LucidWorks Search must be compliant with the JDBC 3 or [JDBC 4 specification](#).

- [API Entry Points](#)
- [Get a List of JDBC Drivers](#)
- [Upload a New JDBC driver](#)
- [Delete a driver](#)
- [Get a List of JDBC 4.0 Compliant Drivers](#)

API Entry Points

/api/collections/collection/jdbcdriivers: [get](#) a list of JDBC drivers or [upload](#) a new one
/api/collections/collection/jdbcdriivers/filename: update, delete, or get file contents
/api/collections/collection/jdbcdriivers/classes: get a list of JDBC 4.0 compliant drivers

Get a List of JDBC Drivers

GET /api/collections/collection/jdbcdriivers

Input

Path Parameters

Key	Description
collection	The collection name

Query Parameters

None.

Input Content

None.

Output

Output Content

Key	Type	Description
filename	string	The driver filenames in a list

Examples

Get a list of all the driver jar files installed in the system.

Input


```
curl http://localhost:8888/api/collections/collection1/jdbcdriver
```

Output

```
[  
  "mysql.jar",  
  "postgresql.jar"  
]
```

[Back to Top](#)

Upload a New JDBC driver

 POST /api/collections/collection/jdbcdriver

You need to upload the JDBC driver using multipart/form-data file upload.

Input

Path Parameters

Key	Description
collection	The collection name

Query Parameters

None.

Input Content

file=driver filename

Output

Output Content

None.

Examples

Upload the mysql.jar file to the system in order to support MySQL.

Input


```
curl -F file=@/path/to/mysql.jar  
http://localhost:8888/api/collections/collection1/jdbcddrivers
```

Output

None.

[Back to Top](#)

Delete a driver

 DELETE /api/collections/collection/jdbcddrivers/filename

Input

Path Parameters

Key	Description
collection	The collection name
filename	The driver filename

Query Parameters

None.

Input Content

None.

Output

Output Content

Key	Type	Description
-----	------	-------------

Examples

Remove MySQL support.

Input


```
curl -X DELETE http://localhost:8888/api/collections/collection1/jdbcddrivers/mysql.jar
```


Output

None.

[Back to Top](#)

Get a List of JDBC 4.0 Compliant Drivers

 GET /api/collections/collection/jdbcdrivers/classes

Input**Path Parameters**

Key	Description
collection	The collection name

Query Parameters

None.

Input Content

None.

Output**Output Content**

None.

Examples

Get a list of the JDBC classes available to datasources.

Input

```
curl http://localhost:8888/api/collections/collection1/jdbcdrivers/classes
```

Output

```
[  
  "com.mysql.jdbc.Driver",  
  "oracle.jdbc.OracleDriver"  
]
```

[Back to Top](#)

FieldTypes

The FieldType API allows creating, updating and deleting field types that are used by LucidWorks Search to define how text found in a field should be processed during indexing. For example, dates should be processed differently from prices as they are generally structured differently in documents. The "date" field would use a field type appropriate for dates, while the "price" field would use a field type appropriate for prices. Many field types are included with LucidWorks and in some cases new ones do not need to be created. However, if the existing list of field types is insufficient for your implementation, this API will allow you to create new field types without manually editing the `schema.xml` configuration file. Alternately, however, the LucidWorks Admin UI includes a [Field Types screen](#) to create, update and delete field types.



Field Types Require Advanced Knowledge of Solr's Schema

LucidWorks uses Solr's `schema.xml` to define field types and fields, and all the functionality contained in Solr is available within LucidWorks Search. That said, FieldType configuration is for advanced users. This API is intended for those who prefer not to edit the `schema.xml` file by hand but who would be entirely comfortable doing so if required.



About Field Type Properties

There are only two properties common to all field types: "class" and "name". Other properties will vary according to the class. Some may have a map of analyzers which may include `char_filters`, `tokenizers`, and/or `token_filters`. Property names (for the class, name, analyzers, etc.) are all strings, even if they are numeric or boolean. They follow the naming convention found in `schema.xml`, which is to say that while most of the properties in LucidWorks Search follow a "under_score" naming convention, field type properties will generally be in "camelCase", meaning that the property names are identical to the attribute names as specified in a valid Solr `schema.xml` file.


- [API Entry Points](#)
- [Get a List of All Field Types](#)
- [Get Details for a Specific Field Type](#)
- [Create a Field Type](#)
- [Update Details for a Specific Field Type](#)
- [Delete a Specific Field Type](#)

API Entry Points

/api/collections/collection/fieldtypes: [get](#) a list of all field types

/api/collections/collection/fieldtypes/fieldtype: [create](#), [update](#), [delete](#), or [get details](#) for a specific field type

Get a List of All Field Types

 GET /api/collections/collection/fieldtypes

Input

Path Parameters

Key	Description
collection	The collection where this field type is available

Query Parameters

None

Output

Output Content

There are only two properties that are common to all FieldTypes; the other properties vary by the class of the field type. All of the classes available in Solr are also available in LucidWorks.

Key	Type	Description
class	string	The implementing class for this field type
name	string	The name of the field type

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/fieldtypes
```

Output

The example below has been truncated for space (indicated by "..." at the beginning and end of the example. The full output will be much longer).


```
...  
{  
  "class": "solr.DateField",
```

```
"name": "pdate",
"omitNorms": "true",
"sortMissingLast": "true"
},
{
  "class": "solr.TextField",
  "name": "text_ws",
  "positionIncrementGap": "100",
  "analyzers": {
    "default": {
      "char_filters": [],
      "tokenizer": {
        "class": "solr.WhitespaceTokenizerFactory"
      },
      "token_filters": []
    }
  }
}
...

```

[Back to Top](#)

Get Details for a Specific Field Type

 GET /api/collections/collection/fieldtypes/fieldtype

Input

Path Parameters

Key	Description
collection	The collection where this field type is available
fieldtype	The name of the field type

Query Parameters

None.

Output

Output Content

There are only two properties that are common to all FieldTypes; the other properties vary by the class of the field type.

Key	Type	Description
class	string	The implementing class for this field type

Key	Type	Description
name	string	The name of the field type

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/fieldtypes/text_en
```

Output


```
{
  "analyzers" : {
    "index" : {
      "token_filters" : [
        {
          "splitOnCaseChange" : "1",
          "catenateWords" : "1",
          "catenateAll" : "0",
          "generateNumberParts" : "1",
          "class" : "solr.WordDelimiterFilterFactory",
          "catenateNumbers" : "1",
          "generateWordParts" : "1"
        },
        {
          "class" : "solr.LowerCaseFilterFactory"
        },
        {
          "class" : "solr.ASCIIFoldingFilterFactory"
        },
        {
          "class" : "com.lucid.analysis.LucidPluralStemFilterFactory",
          "rules" : "LucidStemRules_en.txt"
        }
      ],
      "char_filters" : [],
      "tokenizer" : {
        "class" : "solr.WhitespaceTokenizerFactory"
      }
    },
    "query" : {
      "token_filters" : [
        {
          "ignoreCase" : "true",
          "synonyms" : "synonyms.txt",
          "expand" : "true",
          "class" : "solr.SynonymFilterFactory"
        }
      ],

```

```
{
  {
    "ignoreCase" : "true",
    "class" : "solr.StopFilterFactory",
    "words" : "stopwords.txt"
  },
  {
    "splitOnCaseChange" : "1",
    "catenateWords" : "0",
    "catenateAll" : "0",
    "generateNumberParts" : "1",
    "class" : "solr.WordDelimiterFilterFactory",
    "catenateNumbers" : "0",
    "generateWordParts" : "1"
  },
  {
    "class" : "solr.LowerCaseFilterFactory"
  },
  {
    "class" : "solr.ASCIIFoldingFilterFactory"
  },
  {
    "class" : "com.lucid.analysis.LucidPluralStemFilterFactory",
    "rules" : "LucidStemRules_en.txt"
  }
],
"char_filters" : [],
"tokenizer" : {
  "class" : "solr.WhitespaceTokenizerFactory"
}
},
"positionIncrementGap" : "100",
"name" : "text_en",
"class" : "solr.TextField"
}
```

[Back to Top](#)

Create a Field Type

 POST /api/collections/collection/fieldtypes

Input

Path Parameters

Key	Description
collection	The collection where this field type will be available

Query Parameters

None.

Input Content

See the note about properties at the [top of the page](#).

Output

Output Content

Key	Type	Description
class	string	The implementing class for this field type
name	string	The name of this field type

Other properties will also be listed if set during creation.

Examples

Input

Simple Example

```
curl -H 'Content-type: application/json' -d
'{"class":"solr.TextField","name":"newfieldtype"}'
http://localhost:8888/api/collections/collection1/fieldtypes
```

Example with Analyzers

```
curl -H 'Content-type: application/json' -d '{"name" : "test_field","class" :
"solr.TextField","analyzers":{"default":{"token_filters":[],"char_filters":[],
"tokenizer":{"class" :
"solr.WhitespaceTokenizerFactory"}}},"positionIncrementGap":"100"}'
http://localhost:8888/api/collections/collection1/fieldtypes
```

Output

Simple Example

```
{
  "class":"solr.TextField",
  "name":"newfieldtype"
}
```


Example with Analyzers

```
{
  "class":"solr.TextField",
```

```
"name": "test_field",
"positionIncrementGap": "100",
"analyzers": {
  "default": {
    "char_filters": [],
    "tokenizer": {
      "class": "solr.WhitespaceTokenizerFactory"
    },
    "token_filters": []
  }
}
```

[Back to Top](#)

Update Details for a Specific Field Type

 PUT /api/collections/collection/fieldtypes/fieldtype

Input

Path Parameters

Key	Description
collection	The collection name
fieldtype	The field type name

Query Parameters

None.

Input Content

See the note about properties at the [top of the page](#).

Properties can be removed from an existing field type by specifying a null value in the PUT request. The `analyzers` attribute is considered an individual attribute, so any changes to any part of the sections within the `analyzers` attribute require sending the entire `analyzers` attribute.

Output

Output Content

None.


Examples

Input


```
curl -X PUT -H 'Content-type: application/json' -d '{"positionIncrementGap": "50"}'  
http://localhost:8888/api/collections/collection1/fieldtypes/text_en
```

[Back to Top](#)

Delete a Specific Field Type

 DELETE /api/collections/collection/fieldtypes/fieldtype

Input

Path Parameters

Key	Description
collection	The collection where this field type is available
fieldtype	The fieldtype name

Query Parameters

None.

Output

Output Content

None.

Examples

Input

```
curl -X DELETE -H 'Content-type: application/json'  
http://localhost:8888/api/collections/collection1/fieldtypes/payloads
```

[Back to Top](#)

Fields

The Fields API allows for accessing, modifying, or adding field definitions to a [Collection](#) schema.

As of LucidWorks Search v2.1, this API will return only static fields by default, and not fields that have been created because of a dynamic field declaration. To view fields created by a dynamic field rule, use a new parameter for this API (`include_dynamic=true`), which will also allow changing a dynamic field to a static field if used in an update request. To view dynamic field declarations, use the [Dynamic Fields](#) API.



Some fields included with LucidWorks Search by default are required for proper functioning of the application. Some, however, are only needed for specific features and could be removed or modified if necessary. For a full description of the LucidWorks Search default fields and their purpose, see the section [Customizing the Field Schema](#).


- [API Entry Points](#)
- [Get a List of Fields and Attributes for a Collection](#)
- [Create a New Field](#)
- [Get Attributes for a Field](#)
- [Update a Field](#)
- [Delete a Field](#)
- [Related Topics](#)

API Entry Points

`/api/collections/collection/fields`: [get](#) all fields and their attributes for a collection or [create](#) a new field.

`/api/collections/collection/fields/name`: [update](#), [delete](#), or [get](#) details for a particular field.

Get a List of Fields and Attributes for a Collection

 GET `/api/collections/collection/fields`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

`include_dynamic=true` will include dynamic fields in the response.

Output

Output Content

Key	Type	Description
name	string	The name of the field. Field names are case sensitive. Currently a field name must consist of only A-Z a-z 0-9 - _
copy_fields	list	A list of field names that this field will be copied to. It is not possible to copy a field to the <code>id</code> field because it is by default the unique key for the index.
default_boost	float	How much to boost the field when it is not explicitly included in a user's query. The key <code>search_by_default</code> must be true to change this setting from 1.0: the settings value reverts to 1.0 when <code>search_by_default</code> is set to false.
default_value	string	A default text value for the field. The text specified with this key will be entered if the field is empty in the document.
dynamic_base	string	If non null, this indicates the name of the dynamic field in the schema that is the basis for this field's existence in the collection. Fields with a value in this attribute will not be returned unless <code>include_dynamic=true</code> is added to the request.
editable	boolean	Defines if an external client can edit the field; this cannot be changed.
facet	boolean	Set to true if the lucid request handler will facet on this field. This setting enables a field's terms (words for text types, or exact value for "string" type) to be returned to the search client. A field must be indexed to be facetable. This setting can be changed without reindexing, as it is used at query time only.
field_type	string	A valid field type defined in <code>schema.xml</code> , which can be created or modified with the FieldType API . The field type setting controls how a field is analyzed. There are many options available, and more can be added by adding a new plugin to the <code>schema.xml</code> file. It is crucial to understand the underlying values for a field in order to correctly set its type. For full text fields, such as "title", "body", or "description", a text field

Key	Type	Description
		<p>type is generally the desired setting so individual words in the text are searchable. There are various text field types, most of which are language-specific. However, when a text field value is to be taken literally as-is (exact match only, or for faceting), the "string" type is likely the right choice.</p> <p>There are also types for numeric data, including double, float, integer, and long (and variants of each suitable for sorting: sdouble, sfloat, sint, and slong). The date field accepts dates in the form "1995-12-31T23:59:59.999Z", with the fractional seconds optional, and trailing "Z" mandatory.</p> <p>If you change a field type, we strongly recommend reindexing.</p>
highlight	boolean	Set to true if the /lucid request handler will highlight snippets of the stored field value when they are returned to the search client. A field must be stored to be highlighted. This setting can be changed without reindexing, as it is used at query time only.
include_in_results	boolean	Set to true if the lucid request handler will include this field in its returned results. A field must be stored to be included in results. This setting can be changed without reindexing, as it is used at query time only.
index_for_autocomplete	boolean	Set to true if this field will be used as a source for autocomplete. This allows terms from this field to be used in creation of an auto-complete index that will be created by default at the time of indexing. All fields selected for use in auto-complete are combined into a single "autocomplete" field for use in search suggestions. If you change this setting, we recommend that you recreate the auto-complete index as described in Auto-Complete of User Queries .
index_for_spellcheck	boolean	Set to true if this field will be used as a source for spellchecking. This allows terms from this field to be used in the creation of a spell check index. All fields selected for use in spell checking are combined into a single "spell" field for use in search suggestions.
indexed	boolean	

Key	Type	Description
		Set to true if this field will be indexed for full text search. An indexed field is searchable on the words (or exact value) as determined by the field type. Unindexed fields are useful to provide the search client with metadata for display. For example, URL may not be a valuable search term, but it is very valuable information to show users in their results list. For performance reasons, a best practice is to index as few fields as necessary to still give users a satisfactory search experience. If you change this setting, you must reindex all documents.
multi_valued	boolean	Set to true if this field will contain multiple values from a document. Enable this if the document could have multiple values for a field, such as multiple categories or authors. We recommend that you reindex all documents after changing this setting.
num_facets	integer	Shows the number of facets that will be displayed if <code>facet</code> is true.
omit_tf	boolean	The <code>omit_tf</code> attribute sets Solr's <code>omitTermFreqAndPositions</code> attribute in the schema. If true , term frequency and position information will not be indexed. Enable this if the number of times a term occurs in a document (term frequency) and the proximity of a term to other terms (position) should not be stored. This may be useful for fields that are indexed but not used for searching. This option should not be enabled for text fields (for example, field type <code>text_en</code>) since it would prevent the proper operation of phrase queries and other proximity operators such as <code>NEAR</code> which depend on position information. This attribute works in conjunction with the <code>omit_positions</code> attribute; see the description of that attribute for valid combinations of the attributes.
omit_positions	boolean	The <code>omit_positions</code> attribute sets Solr's <code>{omitPositions}</code> attribute in the schema. If true , term position information will not be indexed. Enable this if the proximity of a term to other terms should not be stored. This attribute works with the <code>omit_tf</code> attribute in that it would be possible to remove information about term frequency while retaining proximity

Key	Type	Description
		<p>information. There are three possible valid combinations of <code>omit_tf</code> and <code>omit_positions</code>:</p> <ul style="list-style-type: none"> • <code>omit_tf</code> is true and <code>omit_positions</code> is true: This would not store any term frequency or position information for the field • <code>omit_tf</code> is false and <code>omit_positions</code> is true: This would not store term positions information but would store term frequency • <code>omit_tf</code> is false and <code>omit_positions</code> is false: This would store term positions and term frequency
<code>query_time_stopword_handling</code>	boolean	Set to true if the Lucid query parser will intelligently remove stop words at query time. This will require LucidWorks Search to apply the stop word list to queries that use this specific field. This does not enable stop words across the board, only to queries on this field (which may be most useful for 'body' fields, for example).
<code>search_by_default</code>	boolean	Set to true if this field will be copied to the default search field. This requires that all queries search this field when the user has not specifically defined a field in a query.
<code>short_field_boost</code>	string	Valid values are: none to not boost terms found in short fields at all, moderate to boost moderately with LucidWorks <code>LucidSimilarityFactory</code> implementation, or high to use the standard Lucene boost implementation. This relevancy boost compensates for text in short documents that have fewer opportunities for text matches and may otherwise rank lower in results than they should. Use 'moderate' for typical text fields such as the abstract or body of an article. Use 'high' for very short fields like title or keywords. Use 'none' for non-text fields. We strongly recommend that you follow changes to the short field boost with a full reindex.
<code>stored</code>	boolean	Set to true if the original un-analyzed text will be stored. A field can be stored independently of indexing, and made available in the results sent to to

Key	Type	Description
		a search client. Re-indexing is not necessary when changing the stored field flag, though fields in documents will remain as they were when they were originally indexed until they are reindexed.
synonym_expansion	boolean	Set to true if the Lucid query parser should expand synonyms at query time.
term_vectors	boolean	This attribute is for expert use only with Solr's TermVectorComponent . It may help you achieve better highlighting and MoreLikeThis performance at the expense of a larger index. For more information, see http://wiki.apache.org/solr/FieldOptionsByUseCase .
use_for_deduplication	boolean	Set to true if the contents of this field will be used when doing document de-duplication.
use_in_find_similar	boolean	Set to true if this field will be used with the default More Like This request handler and be taken into consideration in find-similar/more-like-this computations. The field must be indexed for it to be used for find-similar. This setting can be changed without reindexing, as it is used at query time only.

Examples

Get a list of all fields for the collection "social":

Input

```
curl http://localhost:8888/api/collections/social/fields
```

Output

```
[
  {
    "name": "fileSize",
    "default_boost": 1.0,
    "term_vectors": false,
    "default_value": null,
    "index_for_autocomplete": false,
    "use_for_deduplication": false,
    "highlight": false,
    "multi_valued": true,
    "stored": true,
```

```
"indexed": true,
"search_by_default": false,
"facet": false,
"editable": true,
"index_for_spellcheck": false,
"synonym_expansion": false,
"short_field_boost": "high",
"include_in_results": false,
"use_in_find_similar": false,
"query_time_stopword_handling": false,
"field_type": "text_en",
"omit_tf": true,
"copy_fields": [],
"dynamic_base": null
},
{
  "name": "email",
  "default_boost": 1.0,
  "term_vectors": false,
  "default_value": null,
  "index_for_autocomplete": false,
  "use_for_deduplication": false,
  "highlight": false,
  "multi_valued": true,
  "stored": true,
  "indexed": true,
  "search_by_default": false,
  "facet": false,
  "editable": true,
  "index_for_spellcheck": false,
  "synonym_expansion": false,
  "short_field_boost": "high",
  "include_in_results": false,
  "use_in_find_similar": false,
  "query_time_stopword_handling": false,
  "field_type": "text_en",
  "omit_tf": true,
  "copy_fields": [],
  "dynamic_base": null
}
]
```


Include dynamic fields in the response:

Input

```
curl http://localhost:8888/api/collections/social/fields?include_dynamic=true
```

[Back to Top](#)

Create a New Field

 POST /api/collections/collection/fields

Input


Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Input Content

 When creating a new field, the default values for some attributes depends on the properties of the `field_type` specified at the time the field is created. These exceptions are noted below.

Key	Type	Required	Default	Description
name	string	Yes	No default	The name of the field. Field names are case sensitive. Currently a field name can only contain: A-Z a-z 0-9 - _
copy_fields	list <string>	No	null	A list of field names that this field should copy to. It is not possible to copy a field to itself because it is by default the unique index.
default_boost	float	No	1.0	How much to boost the field value when explicitly included in a user's search. The <code>search_by_default</code> must be set to <code>true</code> for this setting to have any effect. The default setting from 1.0: the settings when <code>search_by_default</code> is <code>false</code> .
default_value	string	No	null	A default text value for the field. If the field with this key will be entered in the document.
facet	boolean	No	false	Set to true if the <code>/lucid</code> request should facet on this field. Enables a field's text types, or exact value for

Key	Type	Required	Default	Description
				returned to the search client. indexed to be facetable. This changed without reindexing, i time only.
field_type	string	Yes	No default	<p>A valid field type defined in s. type setting controls how a fi are many options available, a added by adding a new plugin file. It is crucial to understand values for a field in order to c For full text fields, such as "ti "description", a text field type desired setting so individual v searchable. There are various most of which are language-s when a text field value is to b (exact match only, or for face type is likely the right choice.</p> <p>There are also types for num double, float, integer, and lor each suitable for sorting: sdo slong). The date field accepts "1995-12-31T23:59:59.999Z seconds optional, and trailing</p> <p>If you change a field type, we reindexing.</p>
highlight	boolean	No	false	Set to true if the lucid reques highlight snippets of the store they are returned to the sear be stored to be highlighted. T changed without reindexing, i time only.
include_in_results	boolean	No	false	Set to true if the lucid reques include this field in its returne must be stored to be includee setting can be changed witho used at query time only.
index_for_autocomplete	boolean	No	false	

Key	Type	Required	Default	Description
				Set to true if this field should be used for autocomplete. This allows to be used in creation of an a that will be created by default indexing. All fields selected for auto-complete are combined "autocomplete" field for use in If you change this setting, we you recreate the auto-comple in Auto-Complete of User Que
index_for_spellcheck	boolean	No	false	Set to true if this field should be used for spellchecking. This allows to be used in the creation of : All fields selected for use in s combined into a single "spell" search suggestions.
indexed	boolean	No	Inherited from the field_type	Set to true if this field should be used for text search. An indexed field words (or exact value) as det type. Unindexed fields are us search client with metadata fi example, URL may not be a v but it is very valuable informa their results list. For performa practice is to index as few fiel still give users a satisfactory : you change this setting, you i documents.
multi_valued	boolean	No	Inherited from the field_type	Set to true if this field will co from a document. Enable this could have multiple values for multiple categories or authors that you reindex all document setting.
num_facets	integer	No	5	Set to the number of facets to displayed if the facet attribu
omit_tf	boolean	No	Inherited from the field_type	The omit_tf attribute sets Sc omitTermFreqAndPositions schema. If true , term freque

Key	Type	Required	Default	Description
				information will not be indexed. The number of times a term occurs (term frequency) and the position of other terms (position) should be stored. This may be useful for fields that are used for searching. This option is enabled for text fields (for example, text_en) since it would prevent the operation of phrase queries and operators such as NEAR which require position information. This attribute works in conjunction with the omit_positions attribute. See the description of that attribute for more information about the attributes.
omit_positions	boolean	No	Inherited from the field_type	<p>The omit_positions attribute controls whether term position information will be stored. Enable this if the proximity of terms should not be stored. Together with the omit_tf attribute in the field_type, it is possible to remove information about term frequency while retaining position information. There are three possible valid combinations of omit_tf and omit_positions:</p> <ul style="list-style-type: none"> omit_tf is true and omit_positions is true: This would not store any term information for this field. omit_tf is false and omit_positions is true: This would not store term frequency information but would store position information. omit_tf is false and omit_positions is false: This would store both term frequency and position information.
query_time_stopword_handling	boolean	No	false	Set to true if the Lucid query engine should intelligently remove stop words from queries. This will require LucidWorks to apply the stop words to queries that use this specific field. To enable stop words across the entire index, queries on this field (which may be 'body' fields, for example).

Key	Type	Required	Default	Description
search_by_default	boolean	No	false	Set to true if this field should be the default search field. This request will search this field when the user has not defined a field in a query.
short_field_boost	string	No	high	Valid values are: none to omit the field, moderate to boost moderate fields, Sweet Spot Similarity to implement the standard Lucene boost implementation, irrelevancy boost to compensate for documents that have fewer matches and may otherwise rank lower than they should. Use 'moderate' for fields such as the abstract or title. Use 'high' for very short fields like keywords. Use 'none' for non-keywords. Use 'irrelevancy boost' to strongly recommend that you do not use the short field boost with a full-text field.
stored	boolean	No	Inherited from the field_type	Set to true if the original unanalyzed content should be stored. A field can be stored for indexing, and made available to a search client. Reindexing documents when changing the stored field in documents will remain as they were originally indexed until they are reindexed.
synonym_expansion	boolean	No	false	Set to true if the lucid query should expand synonyms at query time.
term_vectors	boolean	No	Inherited from the field_type	This attribute is for expert use only. It may be used for better highlighting and MoreLikeThis at the expense of a larger index size. For more information, see http://wiki.apache.org/solr/FieldIndexing .
use_for_deduplication	boolean	No	false	Set to true if the contents of this field should be used when doing document deduplication.
use_in_find_similar	boolean	No	false	Set to true if this field should be included in the default MoreLikeThis request. This request is used into consideration in "find similar" requests. The field must be indexed for this to work.

Key	Type	Required	Default	Description
				"find similar". This setting controls reindexing, as it is used at qu

Field Configuration for Synonyms

Fields must be properly configured for synonyms to work properly. If you expect synonyms to operate on a specific field, the attributes `search_by_default` and `synonym_expansion` must be enabled or you may experience situations where search results do not include all documents which contain the synonym terms. To achieve the broadest application of synonym matching, these settings are particularly important for the "text_all" field, which is configured this way by default. Unless you give a specific field in your query, LucidWorks Search will query for synonym terms only in those fields that are enabled for default search and enabled for synonym expansion.

Output

Output Content

See the section on [input fields](#) for details of the information in each field of the response.

Examples

Input

```
curl -H 'Content-type: application/json' -d '{"name": "my_new_field","default_value":  
"lucid rocks","multi_valued": true,"stored": true,"indexed": true,"facet":  
true,"index_for_spellcheck": true,"synonym_expansion": true,"field_type":  
"text_en","copy_fields":["text_medium","text_all"]}'  
http://localhost:8888/api/collections/collection1/fields
```

Output

```
{  
  "default_boost": 1.0,  
  "field_type": "text_en",  
  "facet": true,  
  "indexed": true,  
  "short_field_boost": "high",  
  "term_vectors": false,  
  "include_in_results": false,  
  "stored": true,  
  "omit_tf": false,  
  "highlight": false,  
  "editable": true,  
  "search_by_default": false,
```

```
"multi_valued": true,
"default_value": "lucid rocks",
"use_for_deduplication": false,
"name": "my_new_field",
"synonym_expansion": true,
"index_for_spellcheck": true,
"index_for_autocomplete": false,
"query_time_stopword_handling": false,
"copy_fields": [
  "text_medium",
  "text_all",
  "spell"
],
"use_in_find_similar": false
}
```

[Back to Top](#)

Get Attributes for a Field

 GET /api/collections/collection/fields/name

Input

Path Parameters

Key	Description
collection	The collection name.
name	The field name.

Query Parameters

include_dynamic=true will return attributes for a dynamic field.

Input Content

Output

Output Content

A JSON map of keys to values. For a list of keys, see [GET: Output Content](#).

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/fields/my_new_field
```

Output

```
{
  "default_boost": 1.0,
  "field_type": "text_en",
  "facet": true,
  "indexed": true,
  "short_field_boost": "high",
  "term_vectors": false,
  "include_in_results": false,
  "stored": true,
  "omit_tf": false,
  "highlight": false,
  "editable": true,
  "search_by_default": false,
  "multi_valued": true,
  "default_value": "lucid rocks",
  "use_for_deduplication": false,
  "name": "my_new_field",
  "synonym_expansion": true,
  "index_for_spellcheck": true,
  "index_for_autocomplete": false,
  "query_time_stopword_handling": false,
  "copy_fields": [
    "text_medium",
    "text_all",
    "spell"
  ],
  "use_in_find_similar": false
}
```


Get details for a dynamic field:

Input

```
curl
http://localhost:8888/api/collections/collection1/fields/my_new_field?include_dynamic=true
```

[Back to Top](#)

Update a Field

 PUT /api/collections/collection/fields/name

Input

Path Parameters

Key	Description
collection	The collection name.
name	The field name.

Query Parameters

`include_dynamic=true` is required to update a dynamic field, and will convert it to a static field.

Input Content

For a list of keys, see [POST: Input Content](#).

Any keys you don't edit will keep their existing values, but updating a specific key will overwrite all prior settings for that key. For example, if updating the `copy_field` attribute, any fields previously named as copy field destinations will be overwritten by the new set of copy fields defined.

Output

Output Content

None

Examples

Edit the "my_new_field" field so that it's no longer multi-valued. Also change the default value to "lucid really rocks" and remove it from consideration for spellcheck:

Input

```
curl -X PUT -H 'Content-type: application/json' -d '{"default_value":"lucid really rocks","multi_valued":false,"index_for_spellcheck":false}'
http://localhost:8888/api/collections/collection1/fields/my_new_field
```

Output

None.

[Back to Top](#)

Delete a Field

 DELETE /api/collections/collection/fields/name

Note that deleting a field only removes it as an option for new documents; existing documents will retain this field, even after it's been deleted. Also, listing all fields in the collection will still show the field after it's been deleted.

Input

Path Parameters

Key	Description
collection	The collection name.
name	The field name.

Output

Output Content

None.

Examples

Delete the "my_new_field" field.

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/fields/my_new_field
```

Output

None.

Related Topics


- [Customizing the Field Schema](#)
- [Reindexing Content](#)
- [FieldTypes](#)

[Back to Top](#)

Dynamic Fields

Dynamic fields are those which are not explicitly defined, but are created during indexing based on some criteria, such as a prefix or suffix on the field name. For example, a data set may have a number of fields which end in "_b". Using `dynamicField` functionality in Solr, it's possible to add the content of those fields to the index without having to specify each one of them in the `schema.xml` file. The Dynamic Fields API allows for accessing, modifying, or adding dynamic field definitions to a [Collection](#) schema.

Dynamic fields cannot be used for faceting, highlighting, de-duplication, or MoreLikeThis, unlike explicit fields. However, a dynamic field can be converted to a explicit field with the [Fields API](#), at which point those attributes can be enabled and the field used for those features.

 By default, LucidWorks Search includes a dynamic field declaration of `*`. This rule allows the LucidWorks Search crawlers to add fields as needed while processing crawled documents. If not using the crawlers, or are sure of how documents will be parsed by the crawlers, this rule could be removed. See also [Customizing the Field Schema](#) for more information on default dynamic rules.


- [API Entry Points](#)
- [Get a List of Dynamic Fields and Attributes for a Collection](#)
- [Create a New Dynamic Field](#)
- [Get Attributes for a Dynamic Field](#)
- [Update a Dynamic Field](#)
- [Delete a Dynamic Field](#)

API Entry Points

`/api/collections/collection/dynamicfields`: [get](#) all dynamic fields and their attributes for a collection or [create](#) a new dynamic field.

`/api/collections/collection/dynamicfields/name`: [update](#), [delete](#), or [get](#) details for a particular dynamic field.

Get a List of Dynamic Fields and Attributes for a Collection

 GET `/api/collections/collection/dynamicfields`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Type	Description
name	string	The name of the dynamic field. Dynamic field names are case sensitive. Currently a field name must consist of only A-Z, a-z, 0-9, - or _, and either begin or end (but not both) with *. Examples of legal names include <code>attr_*</code> , <code>*_t</code> or <code>*</code> .
copy_fields	list <string>	A list of field names that this field will be copied to.
field_type	string	<p>A valid field type defined in <code>schema.xml</code>, which can be created or modified with the FieldType API. The field type setting controls how a field is analyzed. There are many options available, and more can be added by adding a new plugin to the <code>schema.xml</code> file. It is crucial to understand the underlying values for a field in order to correctly set its type. For full text fields, a text field type is generally the desired setting so individual words in the text are searchable. There are various text field types, most of which are language-specific. However, when a text field value is to be taken literally as-is (exact match only, or for faceting), the "string" type is likely the right choice. There are also types for numeric data, including double, float, integer, and long (and variants of each suitable for sorting: <code>sdouble</code>, <code>sfloat</code>, <code>sint</code>, and <code>slong</code>). The date field accepts dates in the form "1995-12-31T23:59:59.999Z", with the fractional seconds optional, and trailing "Z" mandatory.</p> <p>If you change a field type, we strongly recommend reindexing.</p>
index_for_autocomplete	boolean	Set to true if these fields will be used as a source for autocomplete. This allows terms from these fields to be used in creation of an auto-complete index that will be created by default at the time of indexing. All fields selected for use in auto-complete are combined into a single "autocomplete"

Key	Type	Description
		field for use in search suggestions. If you change this setting, we recommend that you recreate the auto-complete index as described in Auto-Complete of User Queries .
index_for_spellcheck	boolean	Set to true if these fields will be used as a source for spellchecking. This allows terms from these fields to be used in the creation of a spell check index. All fields selected for use in spell checking are combined into a single "spell" field for use in search suggestions.
indexed	boolean	Set to true if these fields will be indexed for full text search. An indexed field is searchable on the words (or exact value) as determined by the field type. Unindexed fields are useful to provide the search client with metadata for display. For example, URL may not be a valuable search term, but it is very valuable information to show users in their results list. For performance reasons, a best practice is to index as few fields as necessary to still give users a satisfactory search experience. If you change this setting, you must reindex all documents.
multi_valued	boolean	Set to true if these fields will be a 'multi_valued' field. Enable this if the document could have multiple values for a field, such as multiple categories or authors. We recommend that you reindex all documents after changing this setting.
omit_tf	boolean	The <code>omit_tf</code> attribute sets Solr's omitTermFreqAndPositions attribute in the schema. If true , term frequency and position information will not be indexed. Set to true this if the number of times a term occurs in a document (term frequency) and the proximity of a term to other terms (position) should NOT be stored. This may be useful for fields that are indexed but not used for searching. This option should not be enabled for text fields (for example, field type <code>text_en</code>) since it would prevent the proper operation of phrase queries and other proximity operators such as NEAR which depend on position information. This attribute works in conjunction with the <code>omit_positions</code> attribute; see the description of that attribute for valid combinations of the attributes.
omit_positions	boolean	The <code>omit_positions</code> attribute sets Solr's omitPositions attribute in the schema. If true , term position information will not be indexed. Set to true this if the proximity of a term

Key	Type	Description
		<p>to other terms should NOT be stored. This attribute works with the <code>omit_tf</code> attribute in that it would be possible to remove information about term frequency while retaining proximity information. There are three possible valid combinations of <code>omit_tf</code> and <code>omit_positions</code>:</p> <ul style="list-style-type: none">• <code>omit_tf</code> is true and <code>omit_positions</code> is true: This would not store any term frequency or position information for the field• <code>omit_tf</code> is false and <code>omit_positions</code> is true: This would not store term positions information but would store term frequency• <code>omit_tf</code> is false and <code>omit_positions</code> is false: This would store term positions and term frequency
stored	boolean	Set to true if the original unanalyzed text will be stored. The fields can be stored independently of indexing, and made available in the results sent to to a search client. Reindexing is not necessary when changing the stored field flag, though fields in documents will remain as they were when they were originally indexed until they are reindexed.
term_vectors	boolean	This attribute is for expert use only with Solr's TermVectorComponent . It may help you achieve better highlighting and MoreLikeThis performance at the expense of a larger index. For more information, see http://wiki.apache.org/solr/FieldOptionsByUseCase .

Examples

Get a list of all dynamic fields for the default LucidWorks collection "collection1":

Input

```
curl http://localhost:8888/api/collections/collection1/dynamicfields
```

Output

```
[
{
  "field_type":"string",
  "multi_valued":true,
  "indexed":true,
  "name":"attr_*",
```

```
"term_vectors":false,
"index_for_spellcheck":false,
"index_for_autocomplete":false,
"omit_tf":true,
"stored":true,
"copy_fields":[ ],
"omit_positions":true
},
{
  "field_type":"date",
  "multi_valued":false,
  "indexed":true,
  "name":"*_dt",
  "term_vectors":false,
  "index_for_spellcheck":false,
  "index_for_autocomplete":false,
  "omit_tf":true,
  "stored":true,
  "copy_fields":[ ],
  "omit_positions":true
}
]
```

[Back to Top](#)

Create a New Dynamic Field

POST /api/collections/collection/dynamicfields

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Input Content



When creating a new dynamic field rule, the default values of some attributes are inherited from the `field_type` specified when the rule is created. These exceptions are noted below.

Key	Type	Required	Default	Description
name	string	Yes	No default	The name of the dynamic field. Dynamic names are case sensitive. Currently, the name must consist of only A-Z, a-z, 0-9, -, or _ <i>either begin or end (but not both) with a dash or underscore</i> . Examples of legal names include <code>att</code>
copy_fields	list <string>	No	null	A list of field names that this field will copy to.
field_type	string	Yes	No default	<p>A valid field type defined in <code>schema.json</code>. The <code>field_type</code> setting controls how a field is analyzed and indexed. There are many options available, and more can be added by adding a new plugin to the <code>schema.json</code> file. It is crucial to understand the underlying behavior of each type in order to correctly index and search values for a field in order to correctly retrieve results.</p> <p>For full text fields, such as "title", "body", "description", a text field type is generally the best choice. The <code>text</code> type is the desired setting so individual words in the field are searchable. There are various text field types, most of which are language-specific. The <code>text</code> type is used when a text field value is to be taken literally (exact match only, or for faceting), the <code>text_phrase</code> type is likely the right choice. There are also types for numeric data, including double, float, and long (and variants of each suitable for storing doubles, floats, ints, and longs). The <code>date</code> type accepts dates in the form "1995-12-31T23:59:59.999Z", with milliseconds optional, and trailing "Z" mandatory.</p> <p>New field types can be created with the <code>FieldMapper</code> API.</p> <p>If you change a field type, we strongly recommend reindexing.</p>
index_for_autocomplete	boolean	No	false	Set to true if these fields should be indexed as a source for autocomplete. This allows these fields to be used in creation of an auto-complete index that will be created at the time of indexing. All fields selected for indexing in auto-complete are combined into a single "autocomplete" field for use in search. If you change this setting, we recommend reindexing.

Key	Type	Required	Default	Description
				you recreate the auto-complete index in Auto-Complete of User Queries .
index_for_spellcheck	boolean	No	false	Set to true if these fields should be used as a source for spellchecking. This allows these fields to be used in the creation of a spellcheck index. All fields selected for use in spellchecking are combined into a single 'index_for_spellcheck' field for use in search suggestions.
indexed	boolean	No	Inherited from the <code>field_type</code>	Set to true if these fields should be indexed for full text search. Indexed fields are searched for the words (or exact value) as determined by the field type. Unindexed fields are useful for the search client with metadata for display. For example, URL may not be a valuable field but it is very valuable information to display in their results list. For performance reasons, best practice is to index as few fields as possible but still give users a satisfactory search experience. If you change this setting, you must reindex all documents.
multi_valued	boolean	No	Inherited from the <code>field_type</code>	Set to true if these fields should be indexed as a 'multi_valued' field. Enable this if the field could have multiple values for a field such as multiple categories or authors. We recommend that you reindex all documents after setting.
omit_tf	boolean	No	Inherited from the <code>field_type</code>	The <code>omit_tf</code> attribute sets Solr's omitTermFreqAndPositions attribute in the schema for this collection. If true , term frequency and position information will not be indexed. Set to true if the number of times a term appears in a document (term frequency) and the relationship between terms (position) should not be stored. This may be useful for fields that are indexed but not used for searching. This should not be enabled for text fields (field type <code>text_en</code>) since it would prevent the proper operation of phrase queries and other proximity operators such as NEAR which rely on position information.

Key	Type	Required	Default	Description
omit_positions	boolean	No	Inherited from the field_type	<p>The <code>omit_positions</code> attribute sets <code>S omitPositions</code> attribute in the schema collection. If true, term position information not be indexed. Enable this if the proximity term to other terms should not be stored. This attribute works with the <code>omit_tf</code> attribute. It would be possible to remove information about term frequency while retaining proximity information. There are three possible combinations of <code>omit_tf</code> and <code>omit_pos</code>:</p> <ul style="list-style-type: none"> • <code>omit_tf</code> is true and <code>omit_pos</code> is false: This would not store any term position information for the field. • <code>omit_tf</code> is false and <code>omit_pos</code> is true: This would not store term frequency information but would store term position information. • <code>omit_tf</code> is false and <code>omit_pos</code> is false: This would store term position and term frequency.
stored	boolean	No	true	Set to true if the original unanalyzed content should be stored. Fields can be stored independently of indexing, and made available in the index to a search client. Reindexing is not required when changing the stored field flag, but the content in documents will remain as they were until they are reindexed.
term_vectors	boolean	No	Inherited from the field_type	<p>This attribute is for expert use only via the <code>TermVectorComponent</code>. It may help with better highlighting and MoreLikeThis at the expense of a larger index. For information, see http://wiki.apache.org/solr/FieldOptions.</p>

Output

Output Content

JSON representation of the created dynamic field.

Examples

Input


```
curl -H 'Content-type: application/json' -d
{
  "name": "*_sh",
  "indexed": true,
  "stored": true,
  "field_type": "text_en"
}' http://localhost:8888/api/collections/collection1/dynamicfields
```

Output

```
{
  "field_type": "text_en",
  "multi_valued": false,
  "indexed": true,
  "name": "*_sh",
  "term_vectors": false,
  "index_for_spellcheck": false,
  "index_for_autocomplete": false,
  "omit_tf": false,
  "stored": true,
  "copy_fields": [ ],
  "omit_positions": false
}
```

[Back to Top](#)

Get Attributes for a Dynamic Field

 GET /api/collections/collection/dynamicfields/name

Input

Path Parameters

Key	Description
collection	The collection name.
name	The dynamic field name.

Query Parameters

None.

Input Content

None.

Output

Output Content

For a list of keys, see [GET: Output Content](#).

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/dynamicfields/attr_*
```

Output

```
{
  "field_type": "string",
  "multi_valued": true,
  "indexed": true,
  "name": "attr_*",
  "term_vectors": false,
  "index_for_spellcheck": false,
  "index_for_autocomplete": false,
  "omit_tf": true,
  "stored": true,
  "copy_fields": [ ],
  "omit_positions": true
}
```

[Back to Top](#)

Update a Dynamic Field

 `PUT /api/collections/collection/dynamicfields/name`

Input

Path Parameters

Key	Description
collection	The collection name.
name	The dynamic field name.

Query Parameters

None.

Input Content

For a list of keys, see [POST: Input Content](#). Any keys you don't edit will keep their existing values.

Output

Output Content

None.

Examples

Edit the "attr_*" dynamic field so that it is multi-valued:

Input

```
curl -X PUT -H 'Content-type: application/json' -d '{"multi_valued":true}'  
http://localhost:8888/api/collections/collection1/dynamicfields/attr_*
```

Output

None. (Check the dynamic field properties to confirm changes.)

[Back to Top](#)

Delete a Dynamic Field

 DELETE /api/collections/collection/dynamicfields/name

Note that deleting a field only removes it as an option for new documents; existing documents will retain this field, even after it's been deleted. Also, listing all fields in the collection will still show the field after it's been deleted.

Input

Path Parameters

Key	Description
collection	The collection name.
name	The dynamic field name.

Query Parameters

none

Input content

None

Output

Output Content

None

Examples

Delete the "*_sh" dynamic field.

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/dynamicfields/*_sh
```

Output

None.

[Back to Top](#)

Filtering Results

The Filtering API allows you to configure instances of filtering-related search components, such as those used to filter search results by [Access Control Lists](#). A filter can then be used to filter search results according to the Users group membership within Active Directory.



The Filtering API creates a `searchComponent` in the collection's `solrconfig.xml` file.

After the `searchComponent` has been created, it should be added to the `/lucid` request handler with the [Search Components API](#).

- [API Entry Points](#)
- [List Existing Filtering Components](#)
- [Get the Configuration of a Single Filtering Component](#)
- [Create a New Filtering Component](#)
- [Update a Filtering Component Instance](#)
- [Delete a Filtering Component](#)

API Entry Points

`/api/collections/collection/filtering/`: [list](#) all filtering instances.

`/api/collections/collection/filtering/instance-name`: [get](#), [update](#), and [delete](#) configuration details about a search component instance.

List Existing Filtering Components

GET `/api/collections/collection/filtering`

Input

Path Parameters

None.

Query Contents

None.

Output

Output Contents

Key	Description
<code>filterer.class</code>	class name that generates the effective filter queries based on tags. Use <code>com.lucid.security.WindowsACLQueryFilterer</code>

Key	Description
filterer.config	hash containing configuration for filterer instance, for <code>com.lucid.security.WindowsACLQueryFilterer</code> , the following keys are supported: <code>fallback_query</code> : Specifies the query to use when AD provided no sid information for the user, by default this is <code>-*:*</code> <code>should_clause</code> : An optional should clause that is used in the top level boolean query that is constructed based on the sid information provided by the Active directory for example if you want to allow all non SMB content to be available to everybody you could use <code>"*:* -data_source_type:smb"</code>
provider.class	class name that reads user groups from Windows ActiveDirectory and builds access tags based on those. Use <code>com.lucid.security.ad.ADACLTagProvider</code>
provider.config	hash containing configuration for provider instance, for <code>com.lucid.security.ad.ADACLTagProvider</code> , the following keys are supported: <code>java.naming.provider.url</code> : specifies the ActiveDirectory LDAP URL <code>java.naming.security.principal</code> : the user ID for accessing ActiveDirectory <code>java.naming.security.credentials</code> : password for the user account accessing ActiveDirectory <code>userFilter</code> : specifies the LDAP search filter for the user in ActiveDirectory. By default, the <code>userFilter</code> is <code>(&(objectclass=user)(userPrincipalName={0}))</code> . <code>groupFilter</code> : specifies the LDAP search filter for the group in ActiveDirectory. By default the <code>groupFilter</code> is <code>(&(objectclass=group))</code> .

Examples

List filterer component configurations in the `social` collection.

Input

```
curl http://localhost:8888/api/collections/social/filtering
```

Output


```
{
  "ad": {
    "provider.class": "com.lucid.security.ad.ADACLTagProvider",
    "filterer.class": "com.lucid.security.WindowsACLQueryFilterer",
    "provider.config": {
      "java.naming.provider.url": "ldap://pdc.domain/",
      "java.naming.security.principal": "ad-user@domain",
    },
  },
}
```



```
"filterer.config":{}  
}  
}
```

[Back to Top](#)

Get the Configuration of a Single Filtering Component

 GET /api/collections/collection/filtering/instance-name

Input

Path Parameters

Key	Description
collection	The collection name
instance	name of the search component instance

Output

Output Content

Key	Description
filterer.class	class name that generates the effective filter queries based on tags. Use <code>com.lucid.security.WindowsACLQueryFilterer</code>
filterer.config	hash containing configuration for filterer instance, for <code>com.lucid.security.WindowsACLQueryFilterer</code> , the following keys are supported: <code>fallback_query</code> : Specifies the query to use when AD provided no sid information for the user, by default this is <code>-*:*</code> <code>should_clause</code> : An optional should clause that is used in the top level boolean query that is constructed based on the sid information provided by the Active directory for example if you want to allow all non SMB content to be available to everybody you could use <code>"*: * -data_source_type:smb"</code>
provider.class	class name that reads user groups from Windows ActiveDirectory and builds access tags based on those. Use <code>com.lucid.security.ad.ADACLTagProvider</code>
provider.config	hash containing configuration for provider instance, for <code>com.lucid.security.ad.ADACLTagProvider</code> , the following keys are supported: <code>java.naming.provider.url</code> : specifies the ActiveDirectory LDAP URL <code>java.naming.security.principal</code> : the user ID for accessing ActiveDirectory <code>java.naming.security.credentials</code> : password for the user account accessing

Key	Description
	ActiveDirectory userFilter: specifies the LDAP search filter for the user in ActiveDirectory. By default, the userFilter is (&(objectclass=user)(userPrincipalName={0})). groupFilter: specifies the LDAP search filter for the group in ActiveDirectory. By default the groupFilter is (&(objectclass=group)).

Examples

Get the filterer configuration for the ad instance in the social collection.

Input

```
curl http://localhost:8888/api/collections/social/filtering/ad
```

Output

```
{
  "filterer.class": "com.lucid.security.WindowsACLQueryFilterer",
  "provider.class": "com.lucid.security.ad.ADACLTagProvider",
  "provider.config":
  {
    "java.naming.provider.url": "ldap://pdc.domain/",
    "java.naming.security.principal": "ad-user@domain",
  }
}
```

[Back to Top](#)

Create a New Filtering Component

 POST /api/collections/collection/filtering

Input

Path Parameters

Key	Description
collection	The collection name
instance	name of the search component instance

Input Content

Hash of configuration values to set. See [Get Output Contents](#) for supported values.

Output

Output Content

None.

Examples

Set the filterer configuration for the `ad` instance in the `social` collection.

```
curl -v -X POST -H "Accept: application/json" -H "Content-Type: application/json" -d
'{"filterer.class":"com.lucid.security.WindowsACLQueryFilterer","provider.class":
"com.lucid.security.ad.ADACLTagProvider","provider.config":{"java.naming.provider.url":
"ldap://10.0.0.50/","java.naming.security.principal":
"user@dc.domain.example","java.naming.security.credentials": "password"}}'
http://localhost:8888/api/collections/social/filtering/ad
```

[Back to Top](#)

Update a Filtering Component Instance

 `PUT /api/collections/collection/filtering/instance`

Input

Path Parameters

Key	Description
collection	The collection name
instance	name of the search component instance

Input Content

Hash of configuration values to set. See [Get Output Contents](#) for supported values.

Output

Output Content

None.


Examples

Set the filterer configuration for the `ad` instance in the `social` collection.

```
curl -v -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d
'{"filterer.class": "com.lucid.security.WindowsACLQueryFilterer","provider.class":
"com.lucid.security.ad.ADACLTagProvider","provider.config":{"java.naming.provider.url":"ld
"password"}}' http://localhost:8888/api/collections/social/filtering/ad
```

[Back to Top](#)

Delete a Filtering Component

 DELETE /api/collections/collection/filtering/instance

Input

Path Parameters

Key	Description
collection	The collection name
instance	name of the search component instance

Output

Output Content

None.

[Back to Top](#)

Search Components

The Search Component API allows you to configure the list of active search components for each particular search handler.

One example of when you'd need to use this API is when you create a new filter with the [Filtering API](#) to support limiting results to only authorized users as defined by Windows Access Control Lists. Once the filter is created, you must add the `searchComponent` name to the `/lucid` request handler with this API.


An understanding of Solr's [search component](#) functionality is helpful when using this API.

- [API Entry Point](#)
- [List Search Components](#)
- [Update Search Components](#)

API Entry Point

`/api/collections/collection/components/list-name?handlerName=/handlerName`: [List](#) or [update](#) search components for a search handler.

List Search Components

 GET `/api/collections/collection/components/list-name?handlerName=/handlerName`

Input

Path Parameters

Key	Description
collection	The collection name.
list-name	The Solr search component list name: all , first , or last . The all list, if present, contains all the search components for your request handler. The first list, if present, appends the additional search components of your search handler to the beginning of the default list the request handler inherits from its parent class. The last list, if present, appends the additional search components of your search handler to the end of the default list the handler inherits from its parent class. You cannot create new lists using this API; you can only work with lists that already exist for your request handler.

Query Parameters

Key	Description
handlerName	The name of request handler; use /lucid for the standard LucidWorks Search query request handler.

Output

Output Content

JSON array of currently configured search components.

Examples

Get the search components for `/lucid` request handler in the `social` collection.

Input


```
curl http://localhost:8888/api/collections/social/components/all?handlerName=/lucid
```

Output

```
["rolefiltering","query","mlt","stats","feedback","highlight","facet","spellcheck","debug"]
```

[Back to Top](#)

Update Search Components

 `PUT /api/collections/collection/components/list-name?handlerName=/handlerName`

Input

Input Content

Path Parameters

Key	Description
collection	The collection name.
list-name	The Solr search component list name: all , first , or last . The all list, if present, contains all the search components for your request handler. The first list, if present, appends the additional search components of your search handler to the beginning of the default list the request handler inherits from its parent class. The last list, if present, appends the additional search components of your search handler to the end of the default list the handler inherits from its parent class. You cannot create new lists using this API; you can only work with lists that already exist for your request handler.

Query Parameters

Key	Description
handlerName	

Key	Description
	The name of the request handler; use /lucid for the standard LucidWorks Search query request handler.

Output

Output Content

None.

Examples

Set the list of search components for `/lucid` request handler in the `social` collection.

Input

```
curl -v -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d
'["adfiltering","query","mlt","stats","feedback","highlight","facet","spellcheck","debug"]
http://localhost:8888/api/collections/social/components/all?handlerName=/lucid
```

Output

None.

[Back to Top](#)

Settings

The Settings API allows for accessing and modifying settings for a given [collection](#). Note that some of the settings listed below cannot be changed by customers with LucidWorks Search hosted on AWS or Azure.


- [API Entry Points](#)
- [Get All Settings for a Collection](#)
- [Get a Particular Setting](#)
- [Update Settings](#)

API Entry Points

`/api/collections/collection/settings`: [get](#) all settings for a collection or [update](#) settings.

`/api/collections/collection/settings/name`: [get](#) a particular setting

Get All Settings for a Collection

 GET `/api/collections/collection/settings`

Input

Path Parameters

Key	Description
collection	The collection name.


Query Parameters

None.

Output

Output Content

Key	Type	Description
auto_complete	boolean	Is true if auto-complete is enabled for use in the LucidWo setting the auto-complete activity to run at regular interv
boosts	Solr function query	Defines the boost to apply to each query. The default boo
boost_recent	boolean	Is true if the lucid request handler should boost recent do
click_enabled	boolean	

Key	Type	Description
		<p>Is true if Click Scoring is enabled. If enabling this feature <code>update_handler_autosoftcommit_*</code> parameters discussed in the update_handler_autosoftcommit_* section for more information about how Click Scoring and LucidWorks Search on-premise only.</p> <div>  There is currently a known issue where Click Scoring in LucidWorks Search is restarted. So, when enabling Click Scoring, the search index is restarted. For details on how to restart, see the section Restarting the Search Index. </div>
<code>click_boost_data</code>	string	The path to Click Scoring boost data. This feature is available only in LucidWorks Search on-premise.
<code>click_boost_field</code>	string	The field name prefix used by Click fields. This feature is available only in LucidWorks Search on-premise.
<code>click_index_location</code>	string	The path to Click boost index (LucidWorks Search on-premise only).
<code>de_duplication</code>	string	<p>In LucidWorks Search, duplicates can be identified by calculating the similarity between documents. Setting <code>de_duplication</code> to on enables de-duplication generally, specific fields need to be selected with the Fields API. If no fields are selected as being the basis for judging duplicate documents, all fields in the document are used as the basis for judging duplicate documents.</p> <p>You can choose from three possible methods of handling duplicates:</p> <ul style="list-style-type: none"> • Off does not identify duplicate documents within the index. • Tag identifies duplicates with a unique tag stored in the index. This approach is recommended, although it may result in duplicate documents appearing in the search results for users. • Overwrite overwrites duplicate documents with the first document found. This approach is only recommended if duplicate detection is working the way you expect. <p>Note that de-duplication does not work properly in SolrCloud.</p>
<code>default_sort</code>	string	Default sort method - valid values are: relevance , date , score , weight , size , type , id , parent , children , score , weight , size , type , id , parent , children .
<code>display_facets</code>	boolean	Is true if the LucidWorks Search default search interface displays facets.
<code>display_fields</code>	string	Defines the fields to use for display of results to users. Prior to 4.0, only "real" fields were supported. This parameter only applies when using the "real" fields also. This parameter only applies when using the "real" fields also.
<code>elevations</code>	JSON map	Defines the documents to be elevated or excluded from relevance. This feature is enabled by default in LucidWorks Search. This API allows you to define elevation definitions that are used for queries. The elevation definitions are stored in the file <code>\$LWS_HOME/conf/solr/cores/collection/conf/</code> .

Key	Type	Description
		<p>The structure of the <code>elevate.xml</code> file is an XML file definir excluded. The API uses a JSON map to write to this file wi</p> <pre> { "elevations": { "query": [{ "doc": "docID", "exclude": false }] } } </pre> <p>It is also possible to define elevations or exclusions using every result to allow you to add it to the elevations list as</p> <p>The list of documents included or excluded is not synchronor <code>elevate.xml</code> file, and then is removed from the index, it c</p>
main_index_ lock_type	string	<p>Defines which Lucene LockFactory to use. When applying The options are:</p> <ul style="list-style-type: none"> • single: using the SingleInstanceLockFactory. This is another process trying to modify the index. • native: using the NativeFSLockFactory. This uses th multiple Solr web applications in the same JVM that • simple: using the SimpleFSLockFactory. This uses a More information is available in the Lucene Wiki: htt
main_index_ max_buffered_docs	integer	<p>Allows setting the <code>maxBufferedDocs</code> parameter in the <code>solr document updates to buffer in memory before they are flu preferred to use the <code>main_index_ram_buffer_size_mb</code>, b reached.</code></p>
main_index_ max_merge_docs	integer	<p>Allows setting the <code>maxMergeDocs</code> parameter in the <code>solrco documents for a single segment. Once this limit is reached as defined by <code>main_index_merge_factor</code> may also occur.</code></p>
main_index_ merge_factor	integer	<p>Allows setting the <code>mergeFactor</code> parameter in the <code>solrcon index is allowed to have before they are coalesced into on most recently opened segment. When that segment is full (defining when a segment is full is done with the <code>main_in settings</code>). When the the <code>main_index_merge_factor</code> is rea section on <code>{{mergeFactor}}</code></code> (https://cwiki.apache.org/confluence/display/solr/IndexCc the Solr Reference Guide for more information.</p>
	integer	

Key	Type	Description
main_index_ ram_buffer_size_mb		Allows setting the <code>ramBufferSizeMb</code> parameter in the <code>solr</code> memory space (in megabytes) document updates can use generally preferable to <code>main_index_max_buffered_docs</code> , reached.
main_index_ term_index_interval	integer	Allows setting the <code>TermIndexInterval</code> for the index and <code>doc</code> regardless of the number of documents. This allows some cause less memory to be used by the <code>IndexReader</code> , but still be used by the <code>IndexReader</code> , but will speed random-access a larger <code>main_index_term_index_interval</code> because query processing and not by term lookup. A system that experiences value for this setting.
main_index_ use_compound_file	boolean	Allows you to set the <code>UseCompoundFile</code> parameter in the <code>solr</code> the multiple index files on disk to a single file. This setting restrict the number of open files allowed per process. See (https://cwiki.apache.org/confluence/display/solr/IndexComp) in the Solr Reference Guide for more information.
main_index_ write_lock_timeout	integer	Defines the maximum time to wait for a write lock.
query_parser	string	Which query parser the lucid search request handler will use
query_time_stopwords	boolean	Is true if stopwords will be removed at query time.
query_time_synonyms	boolean	Is true if synonyms should be added to queries. This will be used in the query request.
search_server_list	list:string	A list of Solr core URLs that the lucid request handler will use for search.
show_similar	boolean	Is true if a "Find Similar" link should be displayed next to
spellcheck	boolean	Is true if the LucidWorks Search default search interface should
stopword_list	list:string	A list of stopwords that will be used if 'query_time_stopwords'
synonym_list	list:string	A list of synonym rules that will be used if 'query_time_synonyms'
unknown_type_handling	string	A valid field type from the core's schema to use for unrecognized (by passing " <code>unknown_type_handling</code> ": "", or choosing "none" the schema.
unsupervised_feedback	boolean	Is true if unsupervised feedback is enabled
unsupervised_ feedback_emphasis	string	

Key	Type	Description
		Defines if unsupervised feedback should emphasize " relev includes nor excludes additional documents, or " recall " w terms to expand the set of documents matched - default i
update_handler_ autocommit_max_docs	integer	Allows setting the <code>maxDocs</code> parameter for <code>autocommit</code> defi defines the number of documents to queue before pushing update_handler_autocommit_max_time parameter in the index.
update_handler_ autocommit_max_time	integer	Allows setting the <code>maxTime</code> parameter for <code>autocommit</code> defi defines the number of milliseconds to wait before pushing update_handler_autocommit_max_docs parameter in the index.
update_handler_ autocommit_open_searcher	boolean	Provides the option to not open a searcher on hard commi keep track of uncommitted updates. The default is true , c
update_handler_ autosoftcommit_max_docs	integer	Allows setting the <code>maxDocs</code> parameter for <code>autosoftcommit</code> commits are used in Solr's Near RealTime search. This set to the index. It works in conjunction with the <code>{{update_h autosoftcommit_max_time}}</code> parameter in that if either li
update_handler_ autosoftcommit_max_time	integer	Allows setting the <code>maxTime</code> parameter for <code>autosoftcommit</code> commits are used in Solr's Near RealTime search. This set documents to the index. It works in conjunction with the u either limit is reached, the documents will be pushed to th
update_server_list	complex	A map that contains two keys: 'server_list' and 'self'. 'serv distributed updates and 'self' should either be null if <i>this</i> s containing <i>this</i> server address if <i>this</i> server will receive up

Examples

Get the existing settings for the collection:

Input

```
curl http://localhost:8888/api/collections/collection1/settings
```

Output

```
{
  "auto_complete": true,
  "boost_recent": true,
  "boosts": [
    "recip(rord(lastModified),1,1000,1000)"
  ]
}
```

```
],
"click_boost_data": "click-data",
"click_boost_field": "click",
"click_enabled": false,
"de_duplication": "off",
"default_sort": "relevance",
"display_facets": true,
"display_fields": [
  "id", "url", "author", "data_source_type", "lastModified",
  "mimeType", "pageCount", "title"],
"elevations": {},
"main_index_lock_type": "native",
"main_index_max_buffered_docs": -1,
"main_index_max_merge_docs": 2147483647,
"main_index_merge_factor": 10,
"main_index_ram_buffer_size_mb": 64.0,
"main_index_term_index_interval": 32,
"main_index_use_compound_file": false,
"main_index_write_lock_timeout": 1000,
"query_parser": "lucid",
"query_time_stopwords": true,
"query_time_synonyms": true,
"search_server_list": [],
"show_similar": true,
"spellcheck": true,
"stopword_list": [
  "a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into",
  "is", "it", "no", "not", "of", "on", "or", "s", "such", "t", "that", "the",
  "their", "then", "there", "these", "they", "this", "to", "was", "will", "with"],
"synonym_list": [
  "lawyer, attorney", "one, 1", "two, 2", "three, 3", "ten, 10",
  "hundred, 100", "thousand, 1000", "tv, television"],
"unknown_type_handling": "text_en",
"unsupervised_feedback": false,
"unsupervised_feedback_emphasis": "relevancy",
"update_handler_autocommit_max_docs": null,
"update_handler_autocommit_max_time": 3600000,
"update_handler_autocommit_open_searcher": true,
"update_handler_autosoftcommit_max_docs": null,
"update_handler_autosoftcommit_max_time": null,
"update_server_list": null
}
```

[Back to Top](#)

Get a Particular Setting

🚩 GET /api/collections/collection/settings/name

Returns a map of settings to values for a given setting.

Input

Path Parameters

Key	Description
collection	The collection name.
name	The name of the setting to return.

Query Parameters

None.

Output

Output Content

The output will depend on the setting chosen. See the section on [listing all settings](#) for details.

Examples

Determine the default parser for the collection.

Input


```
curl 'http://localhost:8888/api/collections/collection1/settings/query_parser'
```

Output:

```
{
  "query_parser": "lucid",
}
```

[Back to Top](#)

Update Settings

 PUT /api/collections/collection/settings

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input Content

Key	Type	Description
auto_complete	boolean	Is true if auto-complete is enabled for use in the LucidWorks Search interface. Setting the auto-complete activity to run at regular intervals.
boosts	Solr function query	Defines the boost to apply to each query. The default boost is 1.0.
boost_recent	boolean	Is true if the lucid request handler should boost recent documents.
click_enabled	boolean	Is true if Click is enabled (LucidWorks Search on-premise only).
click_boost_data	string	The path to Click boost data (LucidWorks Search on-premise only).
click_boost_field	string	The field name prefix used by Click fields (LucidWorks Search on-premise only).
click_index_location	string	The path to Click boost index (LucidWorks Search on-premise only).
de_duplication	string	The valid values are: off , do not de-duplicate; overwrite , overwrite the index that de-duplication does not work properly in SolrCloud mode.
default_sort	string	Default sort method - valid values are: relevance , date , score .
display_facets	boolean	Is true if the LucidWorks Search default search interface displays facets.
display_fields	string	Defines the fields to use for display of results to users. Prior to 3.0, this parameter only applied to "real" fields also. This parameter only applies when using the LucidWorks Search interface.
elevations	JSON map	<p>Defines the documents to be elevated or excluded from results. This API is an internal API that is used for queries. The elevations file is located at <code>\$LWS_HOME/conf/solr/cores/collection/conf/elevate.xml</code>.</p> <p>The structure of the <code>elevate.xml</code> file is an XML file defining documents to be elevated or excluded. The API uses a JSON map to write to this file with the following structure:</p> <pre> { "elevations": { "query": [{ "doc": "docID", "exclude": true }] } } </pre> <p>The <code>exclude</code> attribute allows using the <code>QueryElevationConnector</code> API. The default is false, so if this attribute is not defined for a document, it will be included in the specified query. Multiple documents can be elevated or excluded for a single query. Set <code>exclude</code> to true to exclude the document.</p>

Key	Type	Description
		<p>It is also possible to define elevations or exclusions using every result to allow you to add it to the elevations list as</p> <p>The list of documents included or excluded is not synchronized with the <code>elevate.xml</code> file, and then is removed from the index, it c</p>
main_index_ lock_type	string	<p>Defines which Lucene LockFactory to use. When applying The options are:</p> <ul style="list-style-type: none"> • single: using the SingleInstanceLockFactory. This is another process trying to modify the index. • native: using the NativeFSLockFactory. This uses the multiple Solr web applications in the same JVM that • simple: using the SimpleFSLockFactory. This uses a More information is available in the Lucene Wiki: http://lucene.apache.org/core/3.6.0/docs/locking.html
main_index_ max_buffered_docs	integer	Allows setting the <code>maxBufferedDocs</code> parameter in the <code>solr</code> document updates to buffer in memory before they are flushed. It is preferred to use the <code>main_index_ram_buffer_size_mb</code> , but reached.
main_index_ max_merge_docs	integer	Allows setting the <code>maxMergeDocs</code> parameter in the <code>solrcore</code> documents for a single segment. Once this limit is reached as defined by <code>main_index_merge_factor</code> may also occur.
main_index_ merge_factor	integer	Allows setting the <code>mergeFactor</code> parameter in the <code>solrcore</code> index is allowed to have before they are coalesced into one on most recently opened segment. When that segment is full (defining when a segment is full is done with the <code>main_index</code> settings). When the the <code>main_index_merge_factor</code> is reached section on <code>{{mergeFactor}}</code> (https://cwiki.apache.org/confluence/display/solr/IndexConfiguration) the Solr Reference Guide for more information.
main_index_ ram_buffer_size_mb	integer	Allows setting the <code>ramBufferSizeMb</code> parameter in the <code>solr</code> memory space (in megabytes) document updates can use generally preferable to <code>main_index_max_buffered_docs</code> , reached.
main_index_ term_index_interval	integer	Allows setting the <code>TermIndexInterval</code> for the index and regardless of the number of documents. This allows some cause less memory to be used by the IndexReader, but still be used by the IndexReader, but will speed random-access

Key	Type	Description
		a larger <code>main_index_term_index_interval</code> because query processing and not by term lookup. A system that experiences a large volume of updates may benefit from a higher value for this setting.
<code>main_index_use_compound_file</code>	boolean	Allows you to set the <code>UseCompoundFile</code> parameter in the <code>IndexConfig</code> to merge the multiple index files on disk to a single file. This setting restricts the number of open files allowed per process. See https://cwiki.apache.org/confluence/display/solr/IndexConfig in the Solr Reference Guide for more information.
<code>main_index_write_lock_timeout</code>	integer	Defines the maximum time to wait for a write lock.
<code>query_parser</code>	string	Which query parser the lucid search request handler will use.
<code>query_time_stopwords</code>	boolean	Is true if stopwords will be removed at query time.
<code>query_time_synonyms</code>	boolean	Is true if synonyms should be added to queries. This will only be used in the query request.
<code>search_server_list</code>	list:string	A list of Solr core URLs that the lucid request handler will search.
<code>show_similar</code>	boolean	Is true if a "Find Similar" link should be displayed next to search results.
<code>spellcheck</code>	boolean	Is true if the LucidWorks Search default search interface should display spellcheck suggestions.
<code>stopword_list</code>	list:string	A list of stopwords that will be used if 'query_time_stopwords' is true.
<code>synonym_list</code>	list:string	A list of synonym rules that will be used if 'query_time_synonyms' is true.
<code>unsupervised_feedback</code>	boolean	Is true if unsupervised feedback is enabled
<code>unsupervised_feedback_emphasis</code>	string	Defines if unsupervised feedback should emphasize "relevance" (includes or excludes additional documents), or "recall" (wants to expand the set of documents matched - default is "relevance").
<code>unknown_type_handling</code>	string	A valid field type from the core's schema to use for unrecognized field types.
<code>update_handler_autocommit_max_docs</code>	integer	Allows setting the <code>maxDocs</code> parameter for <code>autoCommit</code> defines the number of documents to queue before pushing to the index.
<code>update_handler_autocommit_max_time</code>	integer	Allows setting the <code>maxTime</code> parameter for <code>autoCommit</code> defines the number of milliseconds to wait before pushing to the index.

Key	Type	Description
update_handler_ autocommit_open_searcher	boolean	Provides the option to not open a searcher on hard commit keep track of uncommitted updates. The default is true , c
update_handler_ autosoftcommit_max_docs	integer	Allows setting the <code>maxDocs</code> parameter for <code>autosoftcommit</code> commits are used in Solr's Near RealTime searching. This them to the index. It works in conjunction with the <code>{{update autosoftcommit_max_time}}</code> parameter in that if either li
update_handler_ autosoftcommit_max_time	integer	Allows setting the <code>maxDocs</code> parameter for <code>autosoftcommit</code> commits are used in Solr's Near RealTime searching. This documents to the index. It works in conjunction with the <code>u either limit is reached, the documents will be pushed to th</code>
update_server_list	complex	A map that contains two keys: 'server_list' and 'self'. 'serv distributed updates and 'self' should either be null if <i>this</i> s containing <i>this</i> server address if <i>this</i> server will receive up

Output

Output Content

None.

Examples

Turn on spell-checking for the collection.

Input

```
curl -X PUT -H 'Content-type: application/json'
-d '{"spellcheck":true}'
http://localhost:8888/api/collections/collection1/settings
```

Output

None. Check properties to confirm changes.

[Back to Top](#)

Caches

The Caches API allows creating, viewing, updating and deleting cache information configured in the `solrconfig.xml` file for the collection.

Caches are used in Solr to store information from the index for faster responses if the index information is needed again. They are associated with a specific instance of a Solr IndexSearcher, and the cached information is valid as long as the IndexSearcher is valid. When a new IndexSearcher is opened, it is usually auto-warmed (pre-populated) with data from the previous cache. The settings on this page help define the size of caches and how they are used to auto-warm a new IndexSearcher.

These are advanced settings, and modifying them should be done with care.


- [API Entry Points](#)
- [Get a List of Caches and Attributes for a Collection](#)
- [Create a Cache and Attributes for a Collection](#)
- [Update Cache Attributes](#)
- [Delete a Cache](#)

API Entry Points

`/api/collections/collection/caches`: [get](#) details of all configured caches or create a new cache.

`/api/collections/collection/chaches/name`: [update](#), [delete](#), or [get](#) details for a specific cache.

Get a List of Caches and Attributes for a Collection

 GET `/api/collections/collection/caches`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Type	Description
name	string	<p>The name of the cache. There are four known caches that are used in most implementations, which correspond to known Solr caches. These are:</p> <ul style="list-style-type: none"> • field_value_cache, which implements Solr's fieldValueCache. This cache is mostly used for faceting. If it is not explicitly set in <code>solrconfig.xml</code>, then it is generated automatically with default values (<code>initial_size = 10</code>, <code>max_size=10000</code>, and <code>autowarm_count = 0</code>). • document_cache, which implements Solr's documentCache. This cache stores Lucene document objects. • filter_cache, which implements Solr's filterCache. This cache stores unordered sets of all documents for a query, and is used by Solr for filter queries (and thus by LucidWorks for the Search Filter functionality). It can also be used for faceting, sorting, or other situations that require the full set of documents that match a query. • query_result_cache, which implements Solr's queryResultCache. This cache stores ordered sets of document IDs from previous searches. <p>A custom cache implementation can also be created and used if a Solr plugin has been created for your application.</p>
class	string	<p>The cache implementation that's used for this cache. There are two available types, <code>solr.LRUCache</code> and <code>solr.FastLRUCache</code>. The other properties for the cache depend on the implementation that is used. See the following table for details of attributes that can be used for each cache implementation.</p> <p>The "LRU" part of the cache class name stands for "Least Recently Used". When an LRU cache fills up, the entry with the oldest last-accessed timestamp is removed to make room for the new entry. Frequently used items tend to remain cached, while less-used items will drop out of cache to be retrieved again later if they are needed. The <code>FastLRUCache</code> is meant to be lock-less, so is well-suited for caches that are hit several times in a request.</p>
regenerator	string	<p>A valid class name which specifies how to re-populate a new cache created by a new Searcher with old cache objects. Usually only declared for custom caches.</p>

`solr.LRUCache` Attributes

Key	Type	Description
size	integer	The maximum number of entries in the cache.
initial_size	integer	The initial number of entries in the cache.
autowarm_count	integer	The number of entries to pre-populate from an old cache, or can be a percentage of the old cache. When a new Searcher is opened, it may be pre-populated with cached objects from the old Searcher. A best practice is to base the autowarm_count on how long it takes to autowarm a new Searcher.

solr.FastLRUCache Attributes

Key	Type	Description
size	integer	The maximum number of entries in the cache.
initial_size	integer	The initial number of entries in the cache.
autowarm_count	integer	The number of entries to pre-populate from an old cache, or can be a percentage of the old cache. When a new Searcher is opened, it may be pre-populated with cached objects from the old Searcher. A best practice is to base the autowarm_count on how long it takes to autowarm a new Searcher.
min_size	integer	When the cache hits its <code>size</code> limit, this allows the cache to try to reduce to this value. The default is <code>0.9 * size</code> .
acceptable_size	integer	When the cache removes old entries, it tries to achieve the <code>min_size</code> . If that is not possible, it tries to achieve <code>acceptable_size</code> instead. The default is <code>0.95 * size</code> .
cleanup_thread	boolean	If set to true , cache cleanup will run in a dedicated separate thread. Very large cache sizes may perform better if they are in a dedicated thread.
show_items	integer	Used to debug what's contained in the cache. Will return the last N accessed items in the Solr Admin UI, using an MBeans Request Handler or JMX. A value of "-1" will display all items in the cache.

Examples

Input


```
curl http://localhost:8888/api/collections/collection1/caches
```

Output

```
[
{
  "initial_size":"512",
  "name":"document_cache",
  "class":"solr.LRUCache",
  "autowarm_count":"0",
  "size":"512"
},
{
  "initial_size":"512",
  "name":"filter_cache",
  "class":"solr.LRUCache",
  "autowarm_count":"256",
  "size":"512"
},
{
  "initial_size":"512",
  "name":"query_result_cache",
  "class":"solr.LRUCache",
  "autowarm_count":"256",
  "size":"512"
},
{
  "acceptable_size":null,
  "initial_size":"10",
  "name":"field_value_cache",
  "class":"solr.FastLRUCache",
  "cleanup_thread":null,
  "autowarm_count":null,
  "show_items":"-1",
  "min_size":null,
  "size":"10000"
}
]
```

[Back to Top](#)

Create a Cache and Attributes for a Collection

 POST /api/collections/collection/caches

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Input Content

Key	Type	Description
name	string	<p>The name of the cache. There are four known caches that are used in most implementations, which correspond to known Solr caches. These are:</p> <ul style="list-style-type: none"> • field_value_cache, which implements Solr's fieldValueCache. This cache is mostly used for faceting. If it is not explicitly set in <code>solrconfig.xml</code>, then it is generated automatically with default values (<code>initial_size = 10</code>, <code>max_size=10000</code>, and <code>autowarm_count = 0</code>). • document_cache, which implements Solr's documentCache. This cache stores Lucene document objects. • filter_cache, which implements Solr's filterCache. This cache stores unordered sets of all documents for a query, and is used by Solr for filter queries (and thus by LucidWorks for the Search Filter functionality). It can also be used for faceting, sorting, or other situations that require the full set of documents that match a query. • query_result_cache, which implements Solr's queryResultCache. This cache stores ordered sets of document IDs from previous searches. <p>A custom cache implementation can also be created and used if a Solr plugin has been created for your application.</p>
class	string	<p>The cache implementation that's used for this cache. There are two available types, <code>solr.LRUCache</code> and <code>solr.FastLRUCache</code>. The other properties for the cache depend on the implementation that is used. See the following table for details of attributes that can be used for each cache implementation.</p> <p>The "LRU" part of the cache class name stands for "Least Recently Used". When an LRU cache fills up, the entry with the oldest last-accessed timestamp is removed to make room for the new entry. Frequently used items tend to remain cached, while less-used items will drop out of cache to be retrieved again later if they are needed. The <code>FastLRUCache</code> is meant to be lock-less, so is well-suited for caches that are hit several times in a request.</p>
regenerator	string	<p>A valid class name which specifies how to re-populate a new cache created by a new Searcher with old cache objects. Usually only declared for custom caches.</p>

solr.LRUCache Attributes

Key	Type	Description
size	integer	The maximum number of entries in the cache.
initial_size	integer	The initial number of entries in the cache.
autowarm_count	integer	The number of entries to pre-populate from an old cache, or can be a percentage of the old cache. When a new Searcher is opened, it may be pre-populated with cached objects from the old Searcher. A best practice is to base the autowarm_count on how long it takes to autowarm a new Searcher. If setting to a percentage, be sure to enclose the value in quotes, such as "autowarm": "50%". Setting to a number of entries does not require quotes.

solr.FastLRUCache Attributes

Key	Type	Description
size	integer	The maximum number of entries in the cache.
initial_size	integer	The initial number of entries in the cache.
autowarm_count	integer	The number of entries to pre-populate from an old cache, or can be a percentage of the old cache. When a new Searcher is opened, it may be pre-populated with cached objects from the old Searcher. A best practice is to base the autowarm_count on how long it takes to autowarm a new Searcher. If setting to a percentage, be sure to enclose the value in quotes, such as "autowarm": "50%". Setting to a number of entries does not require quotes.
min_size	integer	When the cache hits its <code>size</code> limit, this allows the cache to try to reduce to this value. The default is $0.9 * size$.
acceptable_size	integer	When the cache removes old entries, it tries to achieve the <code>min_size</code> . If that is not possible, it tries to achieve <code>acceptable_size</code> instead. The default is $0.95 * size$.
cleanup_thread	boolean	If set to true , cache cleanup will run in a dedicated separate thread. Very large cache sizes may perform better if they are in a dedicated thread.
show_items	integer	Used to debug what's contained in the cache. Will return the last N accessed items in the Solr Admin UI, using an MBeans Request Handler or JMX. A value of "-1" will display all items in the cache.

Output

Output Content

None.

Examples

Input


```
curl -H 'Content-type: application/json' -d
'{"name":"newcache","class":"solr.LRUCache","size":500}'
http://localhost:8888/api/collections/collection1/caches
```

Output

None.

[Back to Top](#)

Update Cache Attributes

 PUT /api/collections/collection/caches/name

Input

Path Parameters

Key	Description
collection	The collection name.
name	The cache name.

Query Parameters

None.

Input Content

See the list of attributes in the section on [creating a cache](#).

Output

Output Content

None.

Examples

Input


```
curl -X PUT -H 'Content-type: application/json' -d '{"autowarm_count":"50%"}'  
http://localhost:8888/api/collections/collection1/caches/newcache
```

Output

None.

[Back to Top](#)

Delete a Cache

 GET /api/collections/collection/caches/name

Input

Path Parameters

Key	Description
collection	The collection name.
name	The cache name.

Query Parameters

None.

Output

Output Content

None.

Examples

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/caches/newcache
```

[Back to Top](#)

Click Scoring

The Click Scoring API allows programmatic access to record click and query "events", which can be used when developing a custom search application to record user click behavior and use it to impact relevance scores following data processing.


This API does not enable or disable Click Scoring, nor kick off log processing. Those actions are covered by the [Settings](#) and [Activities](#) APIs, respectively. More details about Click Scoring are available in the section on the [Click Scoring Relevance Framework](#).

- [API Entry Points](#)
- [Get Recent Events](#)
- [Record User Queries and Clicks](#)

API Entry Points

`/api/collection/collection/click`: [get](#) statistics about recent Click Scoring events (queries or clicks) or [record](#) events.

Get Recent Events

 GET `/api/collections/collection/click`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Type	Description
buffer_size	integer	The number of events in the buffer, if it is enabled.
buffering	boolean	Is true if buffering has been enabled. Buffering allows the Click Scoring process to store events in memory for later writing to the log files. It's meant to be used as a temporary measure and is most useful when log files are being moved during the Click Scoring analysis processes. When this is changed to false events in the buffer are flushed to the log files.

Key	Type	Description
click_count	integer	The number of click events recorded.
logs_count	integer	The number of Click-related log files in use.
query_count	integer	The number of query events recorded.

Examples

Input

```
curl http://localhost:8888/api/collections/collection1/click
```


Output

```
{
  "buffer_size": 0,
  "buffering": false,
  "click_count": 3,
  "logs_count": 1,
  "query_count": 4
}
```

[Back to Top](#)

Record User Queries and Clicks

The PUT request makes an entry into the `click-collection.log` file, which is where all queries and clicks are recorded for later processing.

 PUT `/api/collections/collection/click`

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None.

Output

Output Content

Key	Type	Description
type	string	The type of event being recorded. A value of " q " indicates a query event; a value of " c " indicates a click event.
req	string	A unique request ID. It is used for both types of events.
q	string	Used for query events, the user's query.
qt	long	Used for query events, the time of the query, in milliseconds since epoch.
hits	integer	Used for query events, the number of hits for the query.
u	string	Optionally used for query events, the user ID.
ct	long	Used for click events, the time of the click, in milliseconds since epoch.
doc	string	Used for click events, the document ID that the user selected.
pos	integer	Used for click events, the position of the document in the result list.

Examples

Input

Example Query Event:

```
curl -X PUT -H 'Content-type: application/json' -d
'{"type":"q","req":"34509586770","q":"ipod","qt":1329157201,"hits":545}'
http://localhost:8888/api/collections/collection1/click
```

Example Click Event:

```
curl -X PUT -H 'Content-type: application/json' -d
'{"type":"c","req":"34509598766","ct":1329157350,"doc":"http://www.apple.com","pos":6}'
http://localhost:8888/api/collections/collection1/click
```

Output

Corresponding entries in the `click-collection.log` file:

Q	1329157201	ipod	none~34509586770	545
C	1329157350	none~34509598766	http://www.apple.com	6

[Back to Top](#)

Roles

With LucidWorks Search, roles are used in conjunction with search filters to restrict the set of documents appearing in search results for users to a particular subset of documents in the index based on their username or LDAP-supplied group membership. This API provides a way to programmatically manage roles. Note that this functionality is also available in the Admin UI, where it is called "[Search Filters](#)".

By default, LucidWorks Search contains one role, called DEFAULT, that is allowed to search all documents and the default "admin" user is added to it. The included [Search UI](#) requires a role, but we have configured LucidWorks so if a user is not included in a role, they inherit DEFAULT


- [API Entry Points](#)
- [Retrieve Existing Roles](#)
- [Create a New Role](#)
- [Retrieve an Existing Role](#)
- [Update a Role](#)
- [Delete a Role](#)

API Entry Points

/api/collections/collection/roles: [retrieve](#) existing roles or [add](#) new roles

/api/collections/collection/roles/role: [retrieve](#), [update](#), or [remove](#) roles

Retrieve Existing Roles

 GET /api/collections/collection/roles

Returns a list of role maps. Each map containing the role name, a list of users, a list of groups the role maps to, and a list of filters to apply when a given role reads the index.

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Output

Output Content

Key	Type	Description
name	string	Name of the role.
groups	list:string	A list of groups for this role.
users	list:string	A list of users for this role.
filters	list:string	A list of filters for this role. Note that filters use Solr's filter query capability, which means they also use the default Lucene query parser instead of the include lucid parser.

Examples

Get a list of the existing roles.

Input


```
curl http://localhost:8888/api/collections/collection1/roles
```

Output

```
[
  {
    "groups": [
    ],
    "name": "DEFAULT",
    "filters": [
      "*"
    ],
    "users": [
      "admin"
    ]
  }
]
```

[Back to Top](#)

Create a New Role

 POST /api/collections/collection/roles

Input

Path Parameters

Key	Description
collection	The collection name.

Query Parameters

None

Input Content

Key	Type	Description
name	string	Name of the role.
groups	list:string	A list of groups for this role.
users	list:string	A list of users for this role.
filters	list:string	A list of filters for this role.

Output

Output Content

Key	Type	Description
name	string	Name of the role.
groups	list:string	A list of groups for this role.
users	list:string	A list of users for this role.
filters	list:string	A list of filters for this role.

Examples

Create a new role that only shows documents that have a `status` field of "public", and assign it to the `group1` and `group2` groups, and to `user1`.

Input

```
curl -H 'Content-type: application/json' -d '{"name": "ONLY_PUBLIC", "groups": ["group1", "group2"], "filters": [{"status": "public"}], "users": ["user1"]}' http://localhost:8888/api/collections/collection1/roles
```

Output


```
{
  "name": "ONLY_PUBLIC",
```



```
"groups": [
  "group1",
  "group2"
],
"filters": [
  "status:public"
],
"users": [
  "user1"
]
}
```

[Back to Top](#)

Retrieve an Existing Role

 GET /api/collections/collection/roles/role

Input

Path Parameters

Key	Description
collection	The collection name.
role	The role name.

Query Parameters

None

Input Content

None.

Output

Output Content

Key	Type	Description
name	string	Name of the role.
groups	list:string	A list of groups for this role.
users	list:string	A list of users for this role.
filters	list:string	A list of filters for this role.

Examples

Get the details for the `ONLY_PUBLIC` role:

Input


```
curl http://localhost:8888/api/collections/collection1/roles/ONLY_PUBLIC
```

Output

```
{
  "name": "ONLY_PUBLIC",
  "groups": [
    "group1",
    "group2"
  ],
  "filters": [
    "status:public"
  ],
  "users": [
    "user1"
  ]
}
```

[Back to Top](#)

Update a Role

 `PUT /api/collections/collection/roles/role`

Input

Path Parameters

Key	Description
collection	The collection name.
role	The role name.

Query Parameters

None

Input Content

Key	Type	Description
groups	list:string	A list of groups for this role.
users	list:string	A list of users for this role.
filters	list:string	A list of filters for this role.

Output

Output Content

None.

Examples

Remove `group1` from the `ONLY_PUBLIC` role.

Input


```
curl -X PUT -H 'Content-type: application/json' -d '{"groups": ["group2"]}'  
http://localhost:8888/api/collections/collection1/roles/ONLY_PUBLIC
```

Output

None. (Check properties to confirm changes.)

[Back to Top](#)

Delete a Role

 DELETE `/api/collections/collection/roles/role`

Input

Path Parameters

Key	Description
collection	The collection name.
role	The role name.

Query Parameters

None

Output

Output Content

None.

Examples

Delete the `ONLY_PUBLIC` role.

Input

```
curl -X DELETE http://localhost:8888/api/collections/collection1/roles/ONLY_PUBLIC
```

Output

None.

[Back to Top](#)

Alerts API

[Enterprise Alerts](#) are a way for users to create a search and save it so LucidWorks notifies them when new content has been added to the index. The Alerts API provides a way to create, update, or list user alerts so that you can manage them programmatically.

❗ Even if using the Alerts API to define alerts for users, the Application Server and Outgoing Mail Server must be configured in the [System Settings screen](#) of the Admin UI before alerts will successfully run and notify users.

- [API Entry Points](#)
- [Create an Alert](#)
- [List all Alerts](#)
- [List Alerts for a User](#)
- [View Details of a Single Alert](#)
- [Run an Alert to Get New Results](#)
- [Update the Details of an Alert](#)
- [Delete an Alert](#)

API Entry Points

`/admin/api/alerts/`: [create](#) an alert, or [get](#) a list of all alerts, or [list](#) alerts for a specific user

`/admin/api/alerts/id`: [view](#) details, [update](#) or [delete](#) a single alert

`/admin/api/alerts/id/check`: [run](#) an alert to find new results

✓ API Address

Unlike the other REST APIs with LucidWorks Search, the Alerts API uses the LWE-UI [component](#). If you followed a default installation, this component runs at `http://localhost:8989/`. Modify this URL as needed to match the location of the LWE-UI component if it is not at the default address.

Note also that the syntax of this API is slightly different from others: paths start with 'admin' instead of 'api'.

Create an Alert

📡 POST `/admin/api/alerts`

Input

Path Parameters

None.

Query Parameters

None.

Input Content

Key	Type	Required	Default	Description
name	string	Yes	null	A user-selected name for the alert.
collection	string	Yes	null	The collection to be searched by the alert.
username	string	Yes	null	The name of the user who will receive alerts.
query	string	Yes	null	The query string.
period	32 bit integer	No	null	Frequency to update the alert, in seconds. If period is not specified, the alert will only be checked when the user selects it to be checked. In this sense it becomes more of a "saved search".
email	string	No	null	The email address to send alert messages to. If a period is entered, an email address is required.
filters	JSON map	No	null	Filters are generally made from facets displayed as part of results, but can be made from any field that contains indexed content from documents.

Output

Output Content

Key	Type	Description
id	integer	A unique id for the alert
name	string	A user-selected name for the alert.
collection	string	The collection to be searched by the alert.
username	string	The name of the user who will receive alerts.
query	string	The query string.
checked_at	date	A timestamp of when the alert was last checked.

Key	Type	Description
period	32 bit integer	Frequency the alert will be updated, in seconds. If period is not specified, the alert will only be checked when the user selects it to be checked. In this sense it becomes more of a "saved search".
email	string	The email address to send alert messages to. If a period is entered, an email address is required.
filters	JSON map	Filters are generally made from facets displayed as part of results, but can be made from any field that contains indexed content from documents.

Examples

Create an alert that searches every 24 hours for documents with the word "solr".

Input

```
curl -X POST -H 'Content-type: application/json' -d '{"name":"Solr documents","collection":"collection1","username":"admin","query":"solr","period":86400,"email":"admin@lucidworks.com"}' http://localhost:8989/admin/api/alerts
```

Output

```
{
  "id":3,
  "name":"Solr documents",
  "collection":"collection1",
  "username":"admin",
  "query":"solr",
  "checked_at":"2011-09-09T18:11:37Z"
}
```

[Back to Top](#)

List all Alerts

Get /admin/api/alerts

Input

Path Parameters

None.

Query Parameters

None.

Input Content

None.

Output

Output Content

Key	Description
id	The unique ID of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
period	How frequently this alert will run (if set).
email	The email address that will be sent notifications.
checked_at	The timestamp of the last time the alert was run.
filters	The filters that will be added to the query string.

Examples

Input


```
curl -X GET -H 'Accept: application/json' http://localhost:8989/admin/api/alerts
```

Output

```
{
  "id":1,
  "name":"Solr documents",
  "collection":"collection1",
  "username":"admin",
  "query":"solr",
  "period":86400,
  "email":"test@test.com",
  "checked_at":"2011-09-01T19:49:48Z"
}
```

[Back to Top](#)

List Alerts for a User

 GET /admin/api/alerts?username=username

Input

Path Parameters

None.

Query Parameters

Key	Description
username	The username associated with the alert.

Output

Output Content

Key	Description
id	The unique ID of the alert.
name	The user-defined name of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
period	How frequently this alert will run.
email	The email address that will be sent notifications.
checked_at	The timestamp of the last time the alert was run.
filters	The filters that will be added to the query string.

Examples

List all alerts for the user with a username of "jdoe".

Input

```
curl http://localhost:8989/admin/api/alerts?username=jdoe
```

Output


```
{
  "id":6,
  "name":"Solr PDF Docs"
  "collection":"collection1",
```

```
"username": "jdoe",
"query": "solr",
"period": 86400,
"email": "test@test.com",
"checked_at": "2011-09-01T19:49:48Z"
"filters":
  {"mimeType": ["application/pdf"]}
}
```

Note that this is a multi-valued response.

[Back to Top](#)

View Details of a Single Alert

 GET /admin/api/alerts/id

Input

Path Parameters

Key	Description
id	The unique ID of the alert

Query Parameters

None.

Output

Output Content

Key	Description
id	The unique ID of the alert.
name	The user-defined name of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
period	How frequently this alert will run.
email	The email address that will be sent notifications.
checked_at	The timestamp of the last time the alert was run.

Key	Description
filters	The filters that will be added to the query string. This will only be shown if there are any filters defined.

Examples

Get the details of alert number '1':

Input

```
curl http://localhost:8989/admin/api/alerts/1
```

Output

```
{
  "id":1,
  "name":"Solr documents",
  "collection":"collection1",
  "username":"admin",
  "query":"solr",
  "period":86400,
  "email":"test@test.com",
  "checked_at":"2011-09-01T19:49:48Z"
}
```

[Back to Top](#)

Run an Alert to Get New Results

 PUT /admin/api/alerts/id/check

Input

Path Parameters

Key	Description
id	The unique ID of the alert.

Query Parameters

None.

Output

Output Content

Key	Description
id	The unique ID of the alert.
name	The user-defined name of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
period	How frequently this alert is run.
email	The email address that will be sent notifications.
checked_at	The timestamp of the last run.
filters	The filters that will be added to the query string. This will only be shown if there are any filters defined.
results	The result list, formatted in a JSON block with the responseHeader, response, highlighting, facet counts, and spell check.

Examples

Get the results of alert number 1:

Input

```
curl -X PUT http://localhost:8989/admin/api/alerts/1/check
```

Output


The example below has been shortened - real output will be much longer depending on how many documents there are that match the query.

```
{
  "id":1,
  "name":"Solr documents",
  "collection":"collection1",
  "username":"admin",
  "query":"solr",
  "period":86400,
  "email":"test@test.com",
  "checked_at":"2011-09-01T20:17:19Z",
  "results":{
    "responseHeader":{
      ...
    },
  },
}
```

```
"response":{
  ...
},
"highlighting":{
  ...
},
"facet_counts":{
  ...
},
"spellcheck":{
  ...
}
}
```

[Back to Top](#)

Update the Details of an Alert

 PUT /admin/api/alerts/id

Input

Path Parameters

Key	Description
id	A unique ID of the alert

Query Parameters

None.

Input Content

Only fields that will be updated need to be included in the request.

Key	Description
id	The unique ID of the alert.
name	The user-defined name of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
period	How frequently this alert is run.

Key	Description
email	The email address that will be sent notifications.
filters	The filters that will be added to the query string. This will only be shown if there are any filters defined.

Output

Output Content

Key	Description
id	The unique ID of the alert.
name	The user-defined name of the alert.
collection	The collection that is queried for the alert.
username	The username for the owner of the alert.
query	The search string.
checked_at	The timestamp of the last time the alert was checked.
period	How frequently this alert is run.
email	The email address that will be sent notifications.
filters	The filters that will be added to the query string. This will only be shown if there are any filters defined.

Examples

Change the query for the "City Alert" to "San Francisco".

Input

```
curl -X PUT -H 'Content-type: application/json' -d '{"query":"San Francisco"}'  
http://localhost:8989/admin/api/alerts/3
```


Output

```
{  
  "id":3,  
  "name":"City alert",  
  "collection":"collection1",  
  "username":"smiller",  
  "query":"San Francisco",  
  "period":86400,
```

```
{
  "email": "test@test.com",
  "checked_at": "2011-09-01T19:49:48Z"
}
```

[Back to Top](#)

Delete an Alert

 DELETE /admin/api/alerts/id

Input

Path Parameters

Key	Description
id	The unique ID of the alert

Query Parameters

None.

Output

Output Content

None.

Examples

Delete the City Search alert, number 3.

Input

```
curl -X DELETE http://localhost:8989/admin/api/alerts/3
```

Output

None.

[Back to Top](#)

Users

The Users API allows you to manage users in LucidWorks Search who are manually created in the LucidWorks internal user database. You do not need to use this API if your installation is configured to check user authentication against an LDAP server.



Unlike most of the other APIs in LucidWorks Search, this API uses the same port as the [LWE-UI component](#). If installed with the default port, that would be at `http://localhost:8989/`.

Note that this API also uses a slightly different syntax: instead of starting with 'api', it begins with 'admin'.

- [API Entry Points](#)
- [Get All Users](#)
- [Create a New User](#)
- [Get Information About a User](#)
- [Update a User](#)
- [Remove a User](#)

API Entry Points

`/admin/api/users` [create](#) a user or [get](#) all users

`/admin/api/users/username:` [update](#), [get](#), or [delete](#) a user

Get All Users

🚩 GET `/admin/api/users`

Input

Path Parameters

None

Query Parameters

None

Output

Output Content

Key	Type	Description
username	string	The username for the user. Each username is unique.
email	string	The user's email address.
authorization	string	Either admin or search . Users with 'admin' authorization can access all parts of the LucidWorks Search UI (Admin UI and Search UI); users with 'search' authorization can only access the Search UI. The authorization is case-sensitive and is always all lower-case.
encrypted_password	string	A secure hash of the user's password.

Examples

Get a listing of the existing users in the system.

Input


```
curl http://localhost:8989/admin/api/users
```

Output

```
[
  {
    "username": "tommickle",
    "email": "mickle@here.com",
    "authorization": "admin",
    "encrypted_password":
"$2a$10$LahkxlPD809eG3tThMoZbe.ceQteNcpyEdhmcUELTyBBSgDqmNSQ6 "
  },
  {
    "username": "admin",
    "email": "admin@localhost.com",
    "authorization": "admin",
    "encrypted_password":
"$2a$10$LahkxlPD809eG3tThMoZbe.ceQteNcpyEdhmcUELTyBBSgDqmNSQ6 "
  },
  {
    "username": "suser",
    "email": "john@there.com",
    "authorization": "search",
    "password": "$2a$10$1lTBrGT/1xXW1cay0HeHe.RmcaH3KFZyGKVQUTVV6eRpn1857ncKm"
  }
]
```

[Back to Top](#)

Create a New User

 POST /admin/api/users

Input

Path Parameters

None.

Query Parameters

None.

Input Content

JSON block with all fields.

Key	Type	Description
username	string	The username for the user. Each username must be unique.
email	string	The user's email address.
authorization	string	Either admin or search . Users with 'admin' authorization can access all parts of the LucidWorks Search UI (Admin UI and Search UI); users with 'search' authorization can only access the Search UI. The value for <code>authorization</code> must always be entered in all lower-case letters.
password	string	The user's password.
encrypted_password	string	An alternate to password: a secure hash of the user's password.

Output

Output Content

Key	Type	Description
id	integer	The unique ID for the user.
username	string	The user's username.
email	string	The user's email address.
authorization	string	The authorization for the user.
encrypted_password	string	The user's password.

Examples

Create a new user.

Input


```
curl -H 'Content-type: application/json' -d '{"username": "smiller", "email": "me@here.com", "authorization": "search", "password": "123456"}' http://localhost:8989/admin/api/users
```

Output

```
{
  "username": "smiller",
  "email": "me@here.com",
  "authorization": "search",
  "encrypted_password": "$2a$10$11TBrGT/1xXW1cay0HeHe.RmcaH3KFZyGKVQUTVV6eRpn1857ncKm"
}
```

[Back to Top](#)

Get Information About a User

 GET /admin/api/users/username

Input

Path Parameters

Key	Description
username	The unique username of the user.

Query Parameters

None.

Output

Output Content

Key	Type	Description
id	integer	The unique ID for the user.
username	string	The user's username.
email	string	The user's email address.

Key	Type	Description
authorization	string	The authorization for the user.
encrypted_password	string	The user's password.

Examples

Get information on the `smiller` user.

Input


```
curl http://localhost:8989/admin/api/users/smiller
```

Output

```
{
  "username": "smiller",
  "email": "me@here.com",
  "authorization": "search",
  "password": "$2a$10$11TBrGT/1xXW1cay0HeHe.RmcaH3KFZyGKVQUTVV6eRpn1857ncKm"
}
```

[Back to Top](#)

Update a User

 `PUT /admin/api/users/username`

Input

Path Parameters

Key	Description
username	The unique username of the user to be updated.

Query Parameters

None.

Input Content

Not all attributes need to be passed, but they could if they all need to be updated.

Key	Type	Description
username	string	The username for the user.

Key	Type	Description
email	string	The user's email address.
authorization	string	Either admin or search . Users with 'admin' authorization can access all parts of the LucidWorks Search UI (Search and Admin); users with 'search' authorization can only access the Search UI. The value for authorization must always be entered in all lower-case letters.
password	string	The user's password.
encrypted_password	string	An alternate to password: a secure hash of the user's password.

Output

Output Content

None.

Examples

Change the authorization for the `smiller` username to "admin", and change the password:

Input

```
curl -X PUT -H 'Content-type: application/json' -d
'{"authorization":"admin","password":"batman"}'
http://localhost:8989/admin/api/users/smiller
```

Output

None. Check properties to confirm changes.

[Back to Top](#)

Remove a User

 DELETE /admin/api/users/username

Input

Path Parameters

Key	Description
username	The unique username of the user to remove

Query Parameters

None.

Input content

None.

Output**Output Content**

None.

Examples

Delete the user `smiller`.

Input

```
curl -X DELETE http://localhost:8989/admin/api/users/smiller
```

Output

None.

[Back to Top](#)

SSL Configuration

This functionality is
not available with
LucidWorks Search
on AWS or Azure

The SSL Configuration API allows you to work with some of the LucidWorks Search Secure Socket Layer-related settings. This API does not support configuring the Container-related settings. For more information about configuring the container-related SSL settings, see [Enabling SSL](#).

It is possible to configure LucidWorks Search to allow only mutually authenticated SSL traffic. This feature is controlled with the parameters `auth_require_authorization` and `auth_authorized_clients`. When you set `auth_require_authorization` to true you can control which clients are allowed to access LucidWorks Search by listing the DNs from the certificates in `auth_authorized_clients`.

If you'd like to configure LucidWorks Search to use SSL between the LWE-Core and LWE-Connectors components, see the section [Client Certificates for LWE-Core and Connectors](#).

- [API Entry Points](#)
- [List the Existing SSL Configuration](#)
- [Update SSL Configuration](#)

API Entry Points

`api/config/ssl`: [List](#) or [update](#) the existing SSL configuration.

List the Existing SSL Configuration

🚩 GET `api/config/ssl`

Input

Path Parameters

None.

Query Parameters

None.

Output

Output Content

JSON block with these parameters:

Key	Type	Required	Default	Description
auth_require_authorization	boolean	no		Enforces client authorization (with certificates). When enabled, only clients that are listed in <code>auth_authorized_clients</code> are allowed to access <code>/api</code> and <code>/solr</code> paths.
auth_authorized_clients	array of strings	no	[]	Lists authorized clients (certificate DN), only relevant when <code>auth_require_authorization</code> is set to true.

Examples

List authorized client configuration:

Input


```
curl http://localhost:8888/api/config/ssl
```

Output

```
{
  "auth_require_authorization":false,
  "auth_authorized_clients":["cn=foo"]
}
```

[Back to Top](#)

Update SSL Configuration

 PUT `/api/config/ssl`

Input

Path Parameters

None.

Query Parameters

None.

Input Content

JSON block with these parameters:

Key	Type	Required	Default	Description
auth_require_authorization	boolean	no		Enforces client authorization (with certificates). When enabled, only clients that are listed in <code>auth_authorized_clients</code> are allowed to access <code>/api</code> and <code>/solr</code> paths.
auth_authorized_clients	array of strings	no	<code>[]</code>	Lists authorized clients (certificate DN), only relevant when <code>auth_require_authorization</code> is set to true.

Output

Output Content

None.

Examples

Configure LucidWorks Search so only authorized clients may communicate:

Input

```
curl -X PUT -H 'Content-type: application/json' -d
'{"auth_require_authorization":true,"auth_authorized_clients":["cn=solr"]}'
http://localhost:8888/api/config/ssl
```

Output

None.

[Back to Top](#)

Crawler Status


The crawler status API allows checking the availability of the Connectors component and a short history of communication between the Core component and the Connectors component. Because the API runs in the Core component, and the Connectors may be on a completely separate server or cluster node, this API may be helpful to check communication between the components.

- [API Entry Points](#)
- [Get Connectors Status](#)
- [Related Topics](#)

API Entry Points

`/api/crawlers/status`: [Get](#) the status of the Connectors component

Get Connectors Status

 GET `/api/crawlers/status`

Input

Path Parameters

None.

Query Parameters

None.

Output

Output Content

There are three main areas output from the status request, `clients`, `status` and `messages`.

The `status` section will return "OK" if everything is normal or "WARNING" if the Connectors component is down.

The `clients` section will list the connected clients. If one of these presents an error, then the communication between the components is not operational. The causes for this will be listed in the `messages` section. If `status` in this section is "OK", then communication is normal. For example, if the below is shown as one of the entries under `clients`, then the connection is OK:

```
"http://localhost:8765/connectors/v1/mgr": {
  "status": "OK",
  "messages": [
    { "message": "online",
```

```
"time": "2012-10-29T14:54:17.553Z"
  },
  "clients": {},
  "type": "local"
}
```

If the Connectors component is down, however, this client would have a status of WARNING and some information in the messages section:

```
"http://127.0.0.1:8765/connectors/v1/mgr": {
  "clients": {},
  "messages": [
    {
      "message": "Communication Error (1001) - The connector failed to complete the communication with the server",
      "time": "2012-11-18T20:56:59.774Z"
    }
  ],
  "status": "ERROR",
  "type": "local"
}
```

The messages section will show the time, date, and a history of messages. The messages section in the clients section show the current messages, if any, but the main messages section shows a history of messages. For example, if the Connectors component was stopped and restarted, the message history would look something like:

```
"messages": [
  {
    "message": "Initiating data source loading and back-compat verification...",
    "time": "2012-11-18T20:41:16.859Z"
  },
  {
    "message": "2 data sources loaded and verified.",
    "time": "2012-11-18T20:41:26.161Z"
  },
  {
    "message": "No connection to remote ConnectorManager! Switching to read-only cache (noop).:
      Communication Error (1001) - The connector failed to complete the communication with the server",
    "time": "2012-11-18T20:56:59.774Z"
  },
  {
    "message": "Synchronizing local and remote data sources after re-connect...",
    "time": "2012-11-18T21:02:38.091Z"
  },
  {
    "message": "2 data sources loaded and verified.",
    "time": "2012-11-18T21:02:38.394Z"
  }
]
```

Note the timestamps on this message. First communication was normal, then there was an error in communication, then after the Connectors component was restarted, the communication returned to normal.

Examples

Input

```
curl http://localhost:8888/api/crawlers/status
```

Output

```
{
  "clients": {
    "http://127.0.0.1:8765/connectors/v1/mgr": {
      "clients": {},
      "messages": [
        {
          "message": "online",
          "time": "2012-11-18T20:44:07.026Z"
        }
      ],
      "status": "OK",
      "type": "local"
    },
    "noop": {
      "clients": {},
      "messages": [
        {
          "message": "read-only datasource cache (local)",
          "time": "2012-11-18T20:41:26.162Z"
        },
        {
          "message": "cached 2 datasources",
          "time": "2012-11-18T20:41:26.162Z"
        }
      ],
      "status": "OK",
      "type": "No-op ConnectorManager"
    }
  },
  "messages": [
    {
      "message": "Initiating data source loading and back-compat
      verification...",
      "time": "2012-11-18T20:41:16.859Z"
    }
  ]
}
```

```
    },  
    {  
      "message": "2 data sources loaded and verified.",  
      "time": "2012-11-18T20:41:26.161Z"  
    }  
  ],  
  "status": "OK",  
  "type": "REST"  
}
```

Related Topics

- [Working With LucidWorks Search Components](#)

Example Clients

Example client code demonstrating how to communicate with LucidWorks Search with a variety of programming languages can be found in the `$LWS_HOME/app/examples/` directory of your LucidWorks Search installation.

- [Example Perl Clients](#)
- [Example Python Clients](#)
- [Example .Net Clients](#)



Information for LucidWorks Search in the Cloud Users

Customers using LucidWorks Search hosted on AWS or Azure who are interested in using these clients can download them from our website at [LucidWorks Sample Clients](#).

Example .Net Clients

The `$LWS_HOME/app/examples/csharp` directory contains utilities demonstrating some of the LucidWorks REST API features from C# code. These utilities can be used to assist people in managing their LucidWorks Search installation, or as an example of how to write C# code as part of customer applications that will interact with LucidWorks and Solr.

- [Dependencies](#)
- [Basic Usage](#)

Dependencies

All of these tools require that the "[Json.NET](#)" DLL be installed.

All of these tools assume that the main URL for LucidWorks is "`http://localhost:8888`". If LucidWorks is running elsewhere, please set the `LWE_URL` environment variable appropriately in the shell where you will be using these tools.

All of these tools deal with "collection1" by default. To use a different collection, please set the `LWE_COLLECTION` environment variable appropriately in the shell where you will be using these tools.

Basic Usage



All of these tools can be run without any arguments to see "help" info about their usage.

Get Some basic Info about the collection...

```
info.exe show
info.exe show index_num_docs index_size free_disk_space
```

View, Modify Settings...

```
settings.exe show
settings.exe show boost_recent stopword_list
settings.exe update boost_recent=false stopword_list=a stopword_list=an
stopword_list=the
```

(note that creating a list is done by specifying the same setting key multiple times)

Execute Searches (with optional filters)

```
search.exe "gtk gnome"  
search.exe "gtk -gnome"  
search.exe "+gtk +gnome" "mimeType:text/html"
```


Example Perl Clients

The `$LWS_HOME/app/examples/perl` directory contains utilities demonstrating many of the LucidWorks REST API features from Perl code. These utilities can be used to assist people in managing their LucidWorks Search installation, or as an example of how to write Perl code as part of customer applications that will interact with LucidWorks and Solr.

- [Dependencies](#)
- [Basic Usage](#)
 - [View, Create, Modify, or Delete Collections](#)
 - [Get Basic Information About the Collection](#)
 - [View or Modify Settings](#)
 - [View, Create, Modify, or Delete Data Sources](#)
 - [Manually Start or Stop a Data Source Job](#)
 - [Modify the Schedule of an Existing Data Source](#)
 - [View the Status and Indexing History of Existing Data Sources](#)
 - [View, Create, Modify, Check, or Delete Alerts](#)
 - [View, Create, Modify, or Delete Activities](#)
 - [View the Status and History of Existing Activities](#)
 - [View, Create, Modify, or Delete Fields](#)
 - [View, Create, Modify, or Delete Users](#)
 - [Modify Roles](#)
 - [Pause or Resume All Background Jobs](#)
 - [Execute Searches with Optional Filters](#)
- [Recipes](#)
 - [Indexing Data Sources](#)
 - [Indexing and Activating Filters for Certain Users](#)
 - [Indexing and Activating Click Boosting](#)
 - [Pause and Resume All Background Jobs for Maintenance](#)

Dependencies

All of these tools require that the "[LWP](#)" and "[JSON](#)" Perl modules be installed.

All of these tools assume that the main URL for LucidWorks is "<http://localhost:8888>" and that the URL for the UI is "<http://localhost:8989>"

If LucidWorks is running elsewhere, please set the `LWE_URL` and `LWE_UI_URL` environment variables appropriately in the shell where you will be using these tools.

With the exception of "[collections.pl](#)" (which deals with multiple collections) all of these tools work with "collection1" by default. To use a different collection, please set the `LWE_COLLECTION` environment variable appropriately in the shell where you will be using these tools.

Basic Usage



All of these tools can be run without any arguments to see "help" information about their usage.

View, Create, Modify, or Delete Collections

```
collections.pl show
collections.pl show name=collection1
collections.pl create name=products instance_dir=prod_dir
collections.pl delete name=products
```

Get Basic Information About the Collection

```
info.pl show
info.pl show index_num_docs index_size free_disk_space
```

View or Modify Settings

```
settings.pl show
settings.pl show boost_recent stopword_list
settings.pl update boost_recent=false stopword_list=a stopword_list=an
stopword_list=the
```

(Note that you can create a list by specifying the same setting key multiple times.)

View, Create, Modify, or Delete Data Sources

```
ds.pl show
ds.pl show id=74
ds.pl show name=simple
ds.pl create name=simple type=file crawler=lucid.aperture path=/usr/share/gtk-doc/html
ds.pl create name=docs type=file crawler=lucid.aperture path=/usr/share/gtk-doc/html
crawl_depth=100
ds.pl update id=74 crawl_depth=999
ds.pl update name=simple crawl_depth=999
ds.pl update id=74 name=new_name crawl_depth=999
ds.pl delete id=74
ds.pl delete name=simple
ds.pl delete-all YES YES YES
```

Manually Start or Stop a Data Source Job

```
ds.pl start id=74
ds.pl start name=simple
ds.pl stop id=74
ds.pl stop name=simple
```

Modify the Schedule of an Existing Data Source

```
ds.pl schedule id=74 active=true period=60 start_time=2076-03-06T12:34:56-0800
ds.pl schedule id=74 active=true period=60 start_time=now
ds.pl schedule name=simple active=true period=60 start_time=now
```

View the Status and Indexing History of Existing Data Sources

```
ds.pl status
ds.pl status id=74
ds.pl status name=simple
ds.pl history id=74
ds.pl history name=simple
```

View, Create, Modify, Check, or Delete Alerts

```
alerts.pl show
alerts.pl show username=bob
alerts.pl show id=68
alerts.pl create username=bob query=gnome name=gnome_alert
alerts.pl update id=68 period=5
alerts.pl check id=68
alerts.pl delete id=68
```

View, Create, Modify, or Delete Activities

```
activities.pl show
activities.pl show id=68
activities.pl create type=click active=true period=60
start_time=2076-03-06T12:34:56-0800
activities.pl create type=click active=true period=60 start_time=now
activities.pl update id=68 active=true period=300
activities.pl delete id=68
```

View the Status and History of Existing Activities

```
activities.pl status
activities.pl status id=68
activities.pl history id=68
```

View, Create, Modify, or Delete Fields

```
fields.pl show
fields.pl show name=mimeType
fields.pl create name=category field_type=string facet=true
fields.pl update name=category search_by_default=true
fields.pl delete name=category
```

View, Create, Modify, or Delete Users

```
users.pl show
users.pl show username=admin
users.pl create username=jim authorization=admin password=jimpass
users.pl update username=jim authorization=search
users.pl delete username=jim
```

Modify Roles

```
roles.pl show
roles.pl show name=DEFAULT
roles.pl create name=SECRET users=hank users=sam filters=status:secret
roles.pl update name=DEFAULT filters=status:public
roles.pl append name=SECRET users=jim users=bob groups=executives
roles.pl delete name=SECRET users=hank
roles.pl delete name=OLD
```

Pause or Resume All Background Jobs

```
maintenance.pl pause
maintenance.pl pause force
maintenance.pl resume ds=5 ds_sched=5 ds_sched=7 act_sched=9
```

Execute Searches with Optional Filters

```
search.pl "gtk gnome"
search.pl "gtk -gnome"
search.pl "+gtk +gnome" "mimeType:text/html"
```

[Back to Top](#)

Recipes

Indexing Data Sources

1. Start up LucidWorks
2. Create a data source using files on the same server as LucidWorks:

```
ds.pl create name=localdocs type=file crawler=lucid.aperture  
path=/usr/share/gtk-doc/html crawl_depth=100
```

3. Schedule the "localdocs" data source to be indexed every 30 minutes starting now:

```
ds.pl schedule name=localdocs active=true period=1800 start_time=now
```

4. Create a data source using a remote HTTP server:

```
ds.pl create name=solrwiki type=web crawler=lucid.aperture  
url=http://wiki.apache.org/solr/ crawl_depth=1
```

5. Run the "solrwiki" data source once right now:

```
ds.pl start name=solrwiki
```

6. Periodically check the status of your data sources to see when the initial indexing is done (look for "crawl_state"):

```
ds.pl status
```

7. Execute some searches in your browser:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=configuration>
8. Searches can also be executed via the REST API using search.pl:

```
search.pl configuration
```

Indexing and Activating Filters for Certain Users

1. [Start](#) LucidWorks
2. Create a new role HTML_ONLY to restrict some users and groups to only searching for HTML documents

```
roles.pl create name=HTML_ONLY filters=mimeType:text/html
```

3. Create a new search user named jim:

```
users.pl create username=jim password=jimpass authorization=search
```

4. Add "jim" to the list of users with the HTML_ONLY role:

```
roles.pl append name=HTML_ONLY users=jim
```

5. Create a data source of a directory containing HTML files as well as other plain text files:

```
ds.pl create name=simple type=file crawler=lucid.aperture  
path=/usr/share/gtk-doc/html crawl_depth=100
```

6. Run the data source once right now:

```
ds.pl start name=simple
```

7. periodically check the 'status' of your data source to see when the initial indexing is done (look for "crawl_state"):

```
ds.pl status name=simple
```

8. Use your browser to login as the "jim" (with password "jimpass") and execute a search: <http://localhost:8989/collections/collection1/search?search%5Bq%5D=>
9. As you execute various searches you should only see HTML documents (note the "Type" Facet in the right hand navigation column)
10. Click the "Sign Out" link in the upper-right corner of search pages and Log in again as the "admin" user: http://localhost:8989/users/sign_out
11. Execute the same searches as before: <http://localhost:8989/collections/collection1/search?search%5Bq%5D=>
12. As you execute various searches you should now see all documents (note the "Type" Facet in the right hand navigation column)

Indexing and Activating Click Boosting

1. [Start](#) LucidWorks
2. Update your settings to enable click tracking:

```
settings.pl update click_enabled=true
```

3. Create a data source:

```
ds.pl create name=local_click_ds type=file crawler=lucid.aperture  
path=/usr/share/gtk-doc/html crawl_depth=100
```

4. Schedule the data source to be indexed every 30 minutes starting now:

```
ds.pl schedule name=local_click_ds active=true period=1800 start_time=now
```

5. Schedule the click processing activity to run every 10 minutes:

```
activities.pl create type=click active=true period=600 start_time=now
```

6. periodically check the 'status' of your data source to see when the initial indexing is done (look for "crawl_state"):

```
ds.pl status name=local_click_ds
```

7. Execute a search in your browser:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=gnome>
8. As you execute searches and click on results, you should see the documents you click on filter up to the top of those searches as the click processing activity runs every 10 minutes.

Pause and Resume All Background Jobs for Maintenance

1. [Start](#) LucidWorks
2. Update your settings to enable click tracking:

```
settings.pl update click_enabled=true
```

3. Create a data source:

```
ds.pl create name=local_click_ds type=file crawler=lucid.aperture  
path=/usr/share/gtk-doc/html crawl_depth=100
```

4. Schedule the data source to be indexed every 30 minutes starting now:

```
ds.pl schedule name=local_click_ds active=true period=1800 start_time=now
```

5. Schedule the click processing activity to run every 10 minutes:

```
activities.pl create type=click active=true period=600 start_time=now
```

6. Pause all active data source schedules and activities, blocking until any currently running data sources and activities are finished:

```
maintenance.pl pause
```

This command should output something like the following:

```
$ maintenance.pl pause  
De-Activating activity #9:  
http://localhost:8888/api/collections/collection1/activities/9  
De-Activating schedule of ds#5:  
http://localhost:8888/api/collections/collection1/datasources/5/schedule  
Waiting for any currently running Activities to finish...  
...Done!  
Waiting for any currently running DataSources to finish...  
...Done!  
Run this command to resume everything that was de-activated...  
maintenance.pl resume activity=9 ds=5
```

7. Perform whatever maintenance is needed.
8. When you are ready, run the command mentioned in the output of the "Pause" step to resume scheduled data source and activity processing:

```
maintenance.pl resume activity=9 ds=5
```

9. Your data source and click activity will now continue to run on the previously specified schedules.

[Back to Top](#)

Example Python Clients

The `$LWS_HOME/app/examples/python` directory contains utilities demonstrating many of the LucidWorks REST API features from Python code. These utilities can be used to assist people in managing their LucidWorks Search installation, or as an example of how to write Python code as part of customer applications that will interact with LucidWorks and Solr.

- [Dependencies](#)
- [Basic Usage](#)
 - [Get Basic Information About the Collection](#)
 - [View or Modify Settings](#)
 - [View, Create, Modify, or Delete Data Sources](#)
 - [Modify the Schedule of an Existing Data Source](#)
 - [View the Status and Indexing History of an Existing Data Source](#)
 - [View, Create, Modify, or Delete Activities](#)
 - [View the Status and History of Existing Activities](#)
 - [View, Create, Modify, or Delete Fields](#)
 - [View, Create, Modify, or Delete Users](#)
 - [Modify Roles](#)
 - [Execute Searches with Optional Filters](#)
- [Recipes](#)
 - [Indexing Data Sources](#)
 - [Indexing and Activating Filters for Certain Users](#)
 - [Indexing and Activating Click Boosting](#)

Dependencies

All of these tools require that the "[httplib2](#)" library be available.

All of these tools assume that the main URL for LucidWorks is "[http://localhost:8888](#)" and that the URL for the UI is "[http://localhost:8989](#)"

If LucidWorks is running elsewhere, please set the `LWE_URL` and `LWE_UI_URL` environment variables appropriately in the shell where you will be using these tools.

All of these tools work with with "collection1" by default. To use a different collection, please set the `LWE_COLLECTION` Environment variable appropriately in the shell where you will be using these tools.

Basic Usage



All of these tools can be run without any arguments to see "help" information about their usage.

Get Basic Information About the Collection

```
info.py show
info.py show index_num_docs index_size free_disk_space
```

View or Modify Settings

```
settings.py show
settings.py show boost_recent stopword_list
settings.py update boost_recent=false stopword_list=a stopword_list=an
stopword_list=the
```

(Note that you can create a list by specifying the same setting key multiple times.)

View, Create, Modify, or Delete Data Sources

```
ds.py show
ds.py show id=74
ds.py show name=simple
ds.py create name=simple type=file crawler=Lucid.Aperture path=/usr/share/gtk-doc/html
ds.py create name=docs type=file crawler=Lucid.Aperture path=/usr/share/gtk-doc/html
crawl_depth=100
ds.py update id=74 crawl_depth=999
ds.py update name=simple crawl_depth=999
ds.py update id=74 name=new_name crawl_depth=999
ds.py delete id=74
ds.py delete name=simple
```

Modify the Schedule of an Existing Data Source

```
ds.py schedule id=74 active=true period=60 start_time=2076-03-06T12:34:56-0800
ds.py schedule id=74 active=true period=60 start_time=now
ds.py schedule name=simple active=true period=60 start_time=now
```

View the Status and Indexing History of an Existing Data Source

```
ds.py status
ds.py status id=74
ds.py status name=simple
ds.py history id=74
ds.py history name=simple
```

View, Create, Modify, or Delete Activities

```
activities.py show
activities.py show id=68
activities.py create type=click active=true period=60
```

```
start_time=2076-03-06T12:34:56-0800
activities.py create type=click active=true period=60 start_time=now
activities.py update id=68 period=300
activities.py delete id=68
```

View the Status and History of Existing Activities

```
activities.py status
activities.py status id=68
activities.py history id=68
```

View, Create, Modify, or Delete Fields

```
fields.py show
fields.py show name=mimeType
fields.py create name=category field_type=string facet=true
fields.py update name=category search_by_default=true
fields.py delete name=category
```

View, Create, Modify, or Delete Users

```
users.py show
users.py show username=admin
users.py create username=jim authorization=admin password=jimpass
users.py update username=jim authorization=search
users.py delete username=jim
```

Modify Roles

```
roles.py show
roles.py show name=DEFAULT
roles.py create name=SECRET users=hank users=sam filters=status:secret
roles.py update name=DEFAULT filters=status:public
roles.py append name=SECRET users=jim users=bob groups=executives
roles.py delete name=SECRET users=hank
roles.py delete name=OLD
```

Execute Searches with Optional Filters

```
search.py "gtk gnome"
search.py "gtk -gnome"
search.py "+gtk +gnome" "mimeType:text/html"
```

[Back to Top](#)

Recipes

Indexing Data Sources

1. [Start](#) LucidWorks
2. Create a data source using files on the same server as LucidWorks:

```
ds.py create name=localdocs type=file crawler=lucid.aperture  
path=/usr/share/gtk-doc/html crawl_depth=100
```

3. Schedule the "localdocs" data source to be indexed every 30 minutes starting now:

```
ds.py schedule name=localdocs active=true period=1800 start_time=now
```

4. Create a data source using a remote HTTP server:

```
ds.py create name=solrwiki type=web crawler=lucid.aperture  
url=http://wiki.apache.org/solr/ crawl_depth=1
```

5. Schedule the 'solrwiki' data source to be indexed once right now:

```
ds.py schedule name=solrwiki active=true period=0 start_time=now
```

6. Periodically check the 'status' of your data sources to see when the initial indexing is done (look for "crawl_state"):

```
ds.py status
```

7. Execute some searches in your browser:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=configuration>
8. Searches can also be executed via the REST API using search.py:

```
search.py configuration
```

Indexing and Activating Filters for Certain Users

1. [Start](#) LucidWorks
2. Create a new role HTML_ONLY to restrict some users and groups to only searching for HTML documents

```
roles.py create name=HTML_ONLY filters=mimeType:text/html
```

3. Create a new search user named jim:

```
users.py create username=jim password=jimpass authorization=search
```

4. Add "jim" to the list of users with the HTML_ONLY role:

```
roles.py append name=HTML_ONLY users=jim
```

5. Create a data source of a directory containing HTML files as well as other plain text files:

```
ds.py create name=simple type=file crawler=lucid.aperture
path=/usr/share/gtk-doc/html crawl_depth=100
```

6. Run the data source once right now:

```
ds.py start name=simple
```

7. periodically check the 'status' of your data source to see when the initial indexing is done (look for "crawl_state"):

```
ds.py status name=simple
```

8. Use your browser to login as the "jim" (with password "jimpass") and execute a search:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=>
9. As you execute various searches you should only see HTML documents (note the "Type" Facet in the right hand navigation column)
10. Click the "Sign Out" link in the upper-right corner of search pages and Log in again as the "admin" user: http://localhost:8989/users/sign_out
11. Execute the same searches as before:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=>
12. As you execute various searches you should now see all documents (note the "Type" Facet in the right hand navigation column)

Indexing and Activating Click Boosting

1. [Start](#) LucidWorks
2. Update your settings to enabled click tracking:

```
settings.py update click_enabled=true
```

3. Create a data source:

```
ds.py create name=local_click_ds type=file crawler=lucid.aperture
path=/usr/share/gtk-doc/html crawl_depth=100
```

4. Schedule the data source to be indexed every 30 minutes starting now:

```
ds.py schedule name=local_click_ds active=true period=1800 start_time=now
```

5. Schedule the click processing activity to run every 10 minutes:

```
activities.py create type=click active=true period=600 start_time=now
```

6. Periodically check the status of your data source to see when the initial indexing is done (look for "crawl_state"):

```
ds.py status name=local_click_ds
```

7. Execute a search in your browser:
<http://localhost:8989/collections/collection1/search?search%5Bq%5D=gnome>
8. As you execute searches and click on results, you should see the documents you click on filter up to the top of those searches as the click processing activity runs every 10 minutes.

[Back to Top](#)

Advanced Operations Using the REST API

LucidWorks provides the ability to programmatically control administrative functions using the REST API.

To use the REST API, you send a JSON object via an HTTP request. For example, you can use the REST API to dynamically create a field:

```
curl -d '{"name":"new_field", "term_vectors":true, "default_value":"lucid rocks",
"use_for_deduplication":true, "multi_valued":true, "stored":true, "indexed":true,
"search_by_default":false, "facet":true, "index_for_spellcheck":true,
"synonym_expansion":true, "short_field_boost":"moderate", "field_type":"text_en"}' -H
'Content-type: application/json'
http://localhost:8888/api/collections/collection1/fields
```

The request returns a JSON representation of the created object:

```
{
  "default_boost":1.0,
  "field_type":"text_en",
  "facet":true,
  "indexed":true,
  "short_field_boost":"moderate",
  "term_vectors":true,
  "include_in_results":false,
  "stored":true,
  "omit_tf":false,
  "highlight":false,
  "editable":true,
  "search_by_default":false,
  "user_field":true,
  "multi_valued":true,
  "default_value":"lucid rocks",
  "use_for_deduplication":true,
  "name":"new_field",
  "synonym_expansion":true,
  "index_for_spellcheck":true,
  "index_for_autocomplete":false,
  "query_time_stopword_handling":false,
  "copy_fields":["text_medium","text_all","spell"],
  "use_in_find_similar":false}
```

As with all operations against LucidWorks, you have the option to use any HTTP client to perform these operations, as long as they use the proper methods, as documented under [REST API](#).

Perhaps the most common usage of the REST API is to control and monitor data sources and indexing. Consider this example, from [Getting Started Indexing](#). First, you create the data source:

```
curl -H 'Content-type: application/json' -d
'{"crawler": "lucid.aperture", "type": "web", "url": "http://www.lucidimagination.com/", "crawl_
Website"}' http://localhost:8888/api/collections/collection1/datasources
```

The response includes the id value (it is quite long, so it's been truncated here):

```
{
  "add_failed_docs": true,
  "auth": [],
  "bounds": "none",
  "caching": false,
  "category": "Web",
  "collection": "collection1",
  "commit_on_finish": true,
  "commit_within": 900000,
  "crawl_depth": -1,
  "crawler": "lucid.aperture",
  "exclude_paths": [ ],
  "fail_unsupported_file_types": true,
  "id": 3,
  "ignore_robots": false,
  "include_paths": [ ],
  "indexing": true,
  "log_extra_detail": false,
  "mapping": {
    ...
  },
  "max_bytes": 10485760,
  "max_docs": -1,
  "name": "Lucid Website",
  "parsing": true,
  "proxy_host": "",
  "proxy_password": "",
  "proxy_port": -1,
  "proxy_username": "",
  "type": "web",
  "url": "http://www.lucidimagination.com/",
  "verify_access": true,
  "warn_unknown_mime_types": true
}
```

The next step is to tell LucidWorks to start indexing the documents by submitting a job request:

```
curl -X PUT http://localhost:8888/api/collections/collection1/datasources/3/job
```

This request does not return anything, but it does start the index running. To check the status, we send a GET request:


```
curl http://localhost:8888/api/collections/collection1/datasources/3/status
```

This request tells us that the data source is still being indexed:

```
{
  "batch_job": false,
  "crawl_started": "2012-02-06T18:40:12+0000",
  "crawl_state": "RUNNING",
  "crawl_stopped": null,
  "id": 3,
  "job_id": "6",
  "num_access_denied": 0,
  "num_deleted": 0,
  "num_failed": 2,
  "num_filter_denied": 0,
  "num_new": 227,
  "num_not_found": 0,
  "num_robots_denied": 0,
  "num_total": 229,
  "num_unchanged": 0,
  "num_updated": 0
}
```

Another common use for the REST API is to manage users and alerts. In most cases, you will use an LDAP server to manage users, but you also have the option to manage them internally using the REST API. For example, to create a user, you would send a `POST` request:

```
curl -H 'Content-type: application/json' -d
'{"username": "smiller", "email": "me@here.com", "authorization": "search", "password":
"123456"}' http://localhost:8989/admin/api/users
```

You can also create an Enterprise Alert, which notifies the user when there are new results for a search. For example, you can create an alert that sends the new user, `smiller`, notification when there is new data for the search "robot":

```
curl -H 'Content-type: application/json' -d '{"name": "Robot
search", "collection": "collection1", "username": "smiller", "query": "robots", "period": 86400, "e
http://localhost:8989/admin/api/alerts
```



The Users API and Enterprise Alerts do not run on the LWE-Core (Solr) port; they're handled by the [LWE UI component](#), installed by default on port 8989.

The request returns a JSON representation of the object, which includes the ID:

```
{
  "id":1,
  "name":"Robot search",
  "collection":"collection1",
  "username":"smiller",
  "query":"robots",
  "checked_at":"2012-06-07T14:22:54Z",
  "period":86400,
  "email":"me@here.com"
}
```

You can then use that ID to check the status of the alert:

```
curl -X PUT http://localhost:8989/admin/api/alerts/1/check
```

In this case, we have not yet seen any new results:

```
{
  "checked_at": "2012-06-07T14:22:54Z",
  "collection": "collection1",
  "email": "me@here.com",
  "id": 1,
  "name": "Robot search",
  "period": 86400,
  "query": "robots",
  "username": "smiller"
}
```

About LucidWorks

LucidWorks (formerly known as Lucid Imagination) is the trusted name in Search, Discovery and Analytics, delivering the only enterprise-grade embedded search development solution built on the power of the Apache Lucene/Solr open source search project. Founded in 2008, the company initially provided support, consulting services, documentation and training for the Apache Lucene/Solr open source search project.

Within a few years, the LucidWorks team realized the need to add value to the open source search platform by developing an extensive layer of services which made Lucene/Solr secure and easier to use and manage. The company shipped the first version of its flagship product, LucidWorks Search, in 2011, followed by LucidWorks Big Data in May 2012. LucidWorks continues to offer support, documentation, consulting services and training products for Lucene/Solr.

LucidWorks remains committed to giving back to the Apache Lucene/Solr community. Out of the 37 Core Committers to the Apache Lucene/Solr project, 9 individuals work for LucidWorks, making the company the largest supporter of open source search in the industry. Further, LucidWorks hosts the Lucene Revolution, a conference dedicated to sharing ideas and promoting the Apache Lucene/Solr open source search project.

For more information on product and support options for LucidWorks Search, please write to: sales@lucidworks.com or visit our [website](#). Support inquiries can be submitted to our [Support group](#).



[LucidWorks](#)

3800 Bridge Parkway, Suite 101
Redwood City, CA 94065

Tel: 650.353.4057

Fax: 650.525.1365