

LucidWorks Search Installation & Upgrade Guide

2.6.3 Documentation

Table of Contents

How to Use this Documentation	4
Conventions	4
Customers of LucidWorks Search on AWS or Azure	6
Getting Support & Training	7
Installation and Upgrade Guide	8
Migrating from Solr to LucidWorks	8
Installation	9
Upgrading from a Prior Version	35
Migrating from Solr to LucidWorks	45
Upgrade Notes for v2.6	51
Working With LucidWorks Search Components	54
About the Components	54
Configuring the Components	56
Related Topics	56
System Directories and Logs	57
Locating Files and Directories	57
System Logs	59
LucidWorksLogs Collection	61
Related Topics	62
Starting and Stopping LucidWorks Search	63
Starting a Standalone LucidWorks Search Instance	63
Starting SolrCloud-enabled LucidWorks Search Instances	63
Stopping LucidWorks Search (all modes)	65
Starting or Stopping Components Separately	65
Glossary of Terms	66
A	66
B	66
C	66
D	67
F	67
I	68
M	68
N	68
Q	68
R	68
S	69
T	70
W	70
About LucidWorks	71

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](#)

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

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

Custom Connector Guide

- Introduction to Lucid Connector Framework
- How To Create A Connector

Lucid Query Parser

- How the default query parser handles user requests
- Customization options

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
Information	<div> Notes with a blue background are used for information that is important for you to know.</div>

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.

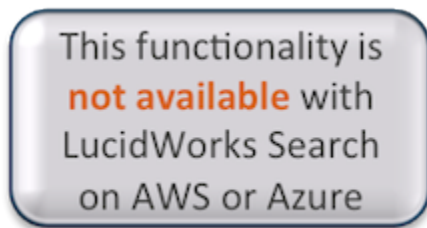
Customers hosted on AWS or Azure should see the section for [Customers of LucidWorks Search on AWS or Azure](#) below.

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-XXXXXXXXX.lucidworks.io/" where XXXXXXXXX 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-9sdff10b.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-9sdff10b.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](#).

Installation and Upgrade Guide

This section covers all aspects of installing and upgrading LucidWorks, as well as some things to consider if you are migrating from a purely Solr installation.

New Installations

- Review the System Requirements.
- **Cluster Installs:** [Plan your cluster](#) and then [install LucidWorks Search and ZooKeeper](#).
- **Standalone (single server) Installs:** Install with [GUI installer](#) or [in console mode](#).
- Review the Known Issues for your version.

Upgrades

- Review the Upgrade Notes to create a plan for the upgrade.
- Follow the instructions described in [Upgrading from a Prior Version](#).
- Review the Known Issues for your version.

Migrating from Solr to LucidWorks

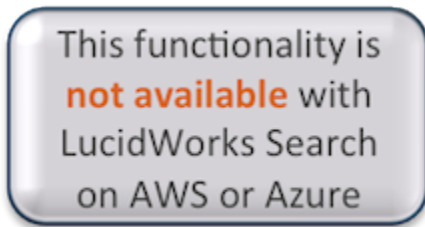
- Review the Requirements.
- Install LucidWorks using a [GUI installer](#) or a [command line installer](#).
- There are some items to consider before migrating your existing Solr configuration files. The section [Migrating from Solr to LucidWorks](#) has the details.
- Review the Known Issues for your version.



Information for LucidWorks Search in the Cloud Users

Many of the installation options are not relevant for LucidWorks Search customers who are hosted on AWS and Azure. However, you will want to review the Upgrade Notes to understand what is new in this release. The LucidWorks Operations team will be in touch when your instances will be upgraded. Please contact your support representative if you have questions.

Installation



There are two main modes of installing LucidWorks:

1. As a single-node standalone application.
 - You can run the installer in **graphical mode**, which guides you through a series of dialog boxes, then installs and configures the software.
 - You can run the installer in **console mode**, which limits the installer's interface to the command line.
2. As a cluster with SolrCloud coordinating search and indexing between nodes.
 - Before starting, make sure you have a plan in place for which nodes will serve which purposes by reviewing the section [Planning a Search Cluster](#).
 - When ready, create a configuration file and start the installations as described in the section [Cluster Installation](#).

Installer File

The installer file is found in the `tar.gz` or `.zip` package you downloaded from LucidWorks. Also included in the package are `README.txt` and `RELEASE_NOTES.txt` files.

For Linux and Mac systems, the install file is called `lucidworks-search-installer-<version>.jar`.

For Windows systems, the install file is called `lucidworks-search-installer-<version>.exe`. On Windows, the installer should be run with Administrator privileges to ensure proper installation. During installation, you will be prompted to install LucidWorks Search as a service. While this enables automatic restart, it is not designed as a monitoring or watchdog service that restarts if a dependent process fails.

Related Topics

- [SolrCloud Cluster Installation](#)
- [Single Server Installation](#)

Single Server Installation with GUI


This functionality is
not available with
LucidWorks Search
on AWS or Azure

Before installing LucidWorks, review the system requirements in the section [System Requirements](#). If you are upgrading your installation of LucidWorks, please see the section [Upgrading from a Prior Version](#) before starting installation.


The graphical installer is simplest for installing LucidWorks Search as a standalone system, meaning not as part of a cluster in SolrCloud mode. If you would like to install Apache ZooKeeper and several instances of LucidWorks Search, please review the section [Cluster Installation](#).

To run the installation wizard, follow these steps:

1. Double-click the installation file (.JAR or .EXE). The Information screen appears showing the version of LucidWorks that will be installed.

 If the installer does not open when you double-click it, open a command shell, make sure that Java 6 or greater is in your path, and launch the installer manually with the command `java -jar <file-name.jar>`.

2. Click **Next**. A list of prerequisites for installing LucidWorks Search appears.
3. Make sure your system meets the specified requirements, then click **Next**. The License Agreements screen appears.
4. Read the license. If you accept its terms, click the button that reads, "I accept the terms of this license agreement."
5. Click **Next**. The next screen asks you to select an installation type. Choose "Standalone installation". This will install LucidWorks Search as a single node.

 If you would like to install ZooKeeper or LucidWorks Search in SolrCloud mode, please see the section [Cluster Installation](#). While it is possible to use the graphical installer for ZooKeeper and SolrCloud mode installation, best practice is to use the configuration mode on the command line.

6. The **Components to Enable** screen appears.

The installer displays a list of LucidWorks Search components and their default addresses. We recommend that you install all components, unless you are working on a custom installation. See [Working With LucidWorks Search Components](#) for more information.

7. Configure the components dialog box to select the LucidWorks Search components and network addresses you want to install.



Remove Default Addresses to Skip Components

If you choose not to install a component, be sure to uncheck the box next to the component name *and* remove its default address (or change it to the location where that component is installed). If the address is not removed or changed, the component will not be installed, but the default address will be entered into the `master.conf` configuration file, which will cause installed components to try to access the skipped component at that address.

8. Click **Next**. The Select Installation Path screen appears. Enter or browse to the directory where LucidWorks Search will be installed. It's best to choose a path that does not contain spaces (i.e., not a path like "c:\program files\" or similar). Paths with spaces will cause the crawlers to load improperly when LucidWorks starts. This will be the location of `$LWS_HOME`, which is referenced throughout this Guide when specifying file paths.

If you are [upgrading a prior version of LucidWorks Search](#), **do not** overwrite your existing installation.

9. Click **Next**. The installer will ask you to confirm the installation location before proceeding.
10. Click **OK**. The LucidWorks System Monitor Screen appears. This screen allows you to opt-in to or opt-out of the LucidWorks System Usage Monitor program, which sends anonymous, encrypted data about your system to LucidWorks. A link on the screen will take you to the LucidWorks website for more information, or you can look in this documentation for the section on LucidWorks System Usage Monitor. Uncheck the box to opt-out of the program, or leave the box checked to opt-in.
11. Click **Next**. The Summary of Choices screen appears.
12. Confirm your installation choices, then click **Next**. The LucidWorks Search installation begins.
13. When the installation is finished, click **Next**. The Start LucidWorks Search screen appears. To start LucidWorks Search immediately, check the Start LucidWorks box. If installing in Windows, you will be prompted to start LucidWorks Search as a service. If you opt not to install as a service, the next screen will ask if you want to start LucidWorks Search.
14. Click **Next**. The installer initiates the LucidWorks Search [start scripts](#). In most installations, these start quickly, but it may take up to one minute for the scripts to complete. The installer allows you to continue while the scripts work in the background.
15. Click **Next**.
16. As a final step, the installer can create an automated installation script that includes the settings you chose that you can use to run unattended installations. To generate an automated installation script, click **Generate an automatic installation script**. Otherwise, click **Done**.

You have now installed LucidWorks Search. If you accepted the default component locations, you can access the [apps landing page](http://localhost:8989/) at <http://localhost:8989/>. Otherwise, you can find the Administrative User Interface at the URL and port you chose in step six. The landing page has these options to get started:

- Quick Start
- LucidWorks Search
- Relevancy Workbench
- Solr Admin

Refer to the README.txt file under the installation root directory for the default password. You can change the default password using the [Users screen](#) in the Admin UI or with an API call described on the Users API page.

Single Server Installation in Console Mode

This functionality is
not available with
LucidWorks Search
on AWS or Azure

If you are installing LucidWorks Search on a computer without a graphical user interface (i.e., a "headless" machine), you can run the installer in "console" mode.

- ⊖ These instructions assume a single-node installation. If you would like to install LucidWorks Search on multiple nodes of a cluster to run in SolrCloud mode, you should review the installation instructions at [Cluster Installation](#).

1. Launch the installer with the command `java -jar <file-name.jar> -console`. (The name of your download may differ from the name displayed here.)

```
$ java -jar lucidworks-search-installer-2.0.jar -console
```

2. Read through the license, pressing "enter" to page through it, then press '1' at the end to accept the license terms.

```
....
General
We may audit and monitor Your use of the Programs.  You agree that this Agreement
is the complete Agreement for the Programs and licenses, and this Agreement
supersedes all prior or contemporaneous Agreements or representations.  If any
term of this Agreement is found to be invalid or unenforceable, the remaining
provisions will remain effective.  This Agreement is governed by the substantive
and procedural laws of California.  You and Lucid agree to submit to the exclusive
jurisdiction of, and venue in, the courts of San Mateo county in California in
any dispute arising out of or relating to this Agreement.
press 1 to accept, 2 to reject, 3 to redisplay

1
```

3. Select each component to install and choose the address and port each component will run on (this is multiple steps).

```
Please select which LucidWorks Search components should be enabled on this
server, or specify the remote address of a component if it will run remotely:
```

```
[x] Run LucidWorks Search Core Locally
```

```
input 1 to select, 0 to deselect:
```

```
1
```

```
Address [http://127.0.0.1:8888]
```

```
[x] Run LucidWorks Search Connectors Locally
```

```
input 1 to select, 0 to deselect:
```

```
1
```

```
Address [http://127.0.0.1:8765]
```

```
[x] Run LucidWorks Search UI Locally
```

```
input 1 to select, 0 to deselect:
```

```
1
```

```
Address [http://127.0.0.1:8989]
```

```
Note: Components will communicate with each other using these addresses, and if
enabled on this machine, will run on the port given in the address. Leave the
address blank to keep a component from being used (locally or remotely)
press 1 to continue, 2 to quit, 3 to redisplay
```

```
1
```

4. After approving the components to install, the installer will check the ports defined and warn if they are currently in use. It's possible to continue with the installer, but LucidWorks Search will not start until the ports are available.

Select the directory location for the install. This will be the base path for `$LWS_HOME`, which is referenced throughout this Guide when discussing configuration and log file locations.

```
Select target path [/Users/cassandra4work/Downloads]
```

```
/Applications/LucidWorks/LucidWorksSearch
```

```
press 1 to continue, 2 to quit, 3 to redisplay
```

```
1
```

5. After defining the installation location, choose whether to opt-in or opt-out of the LucidWorks System Usage Monitor, which will send anonymous, encrypted statistics about your system to LucidWorks.

```
<html>LucidWorks Search product lets you share optional usage statistics with  
LucidWorks to help us improve our products. To find out more about this feature,  
including how to disable it, click <a  
href='http://www.lucidworks.com/lucidworks-system-usage-monitor'>here</a>.</html>
```

```
[x] Enable LucidWorks System Usage Monitor  
input 1 to select, 0 to deselect:
```

6. The installer installs LucidWorks Search, and asks if you want to start the servers. If you intend to use LucidWorks Search in SolrCloud mode, even in a small cluster, you should not start LucidWorks Search with this installer (see the section Using SolrCloud in LucidWorks for more details).

```
2013-09-04 12:51:15,573 INFO panels.InstallPanelConsoleHelper - [ Starting to  
unpack ]  
2013-09-04 12:51:15,577 INFO panels.InstallPanelConsoleHelper - [ Processing  
package: LucidWorks (1/3) ]  
2013-09-04 12:51:15,578 INFO panels.InstallPanelConsoleHelper - [ Processing  
package: LWSshared (2/3) ]  
2013-09-04 12:51:33,623 INFO panels.InstallPanelConsoleHelper - [ Processing  
package: StartLucid (3/3) ]  
2013-09-04 12:51:34,311 INFO panels.InstallPanelConsoleHelper - [ Unpacking  
finished ]  
  
Would you like to start LucidWorks Search now?  
NOTE: LucidWorks Search will be started in the background and may take some time  
to complete loading.  
You may continue on with the installer before it finishes loading.  
Depending on your system hardware, LucidWorks Search may take some time to finish  
loading even after the installer has been closed.  
[x] Start LucidWorks Search  
input 1 to select, 0 to deselect:  
1
```

7. Finally, the installer displays confirmation of the successful installation.

```
Install was successful  
application installed on /Applications/LucidWorks/LucidWorksSearch  
[ Console installation done ]
```

You have now installed LucidWorks Search. If you accepted the default component locations, you can access the [apps landing page](http://localhost:8989/) at <http://localhost:8989/>. Otherwise, you can find the Administrative User Interface at the URL and port you chose in step six. The landing page has these options to get started:

- Quick Start
- LucidWorks Search
- Relevancy Workbench
- Solr Admin

Refer to the README.txt file under the installation root directory for the default password. You can change the default password using the [Users screen](#) in the Admin UI or with an API call described on the Users API page.

Planning a Search Cluster

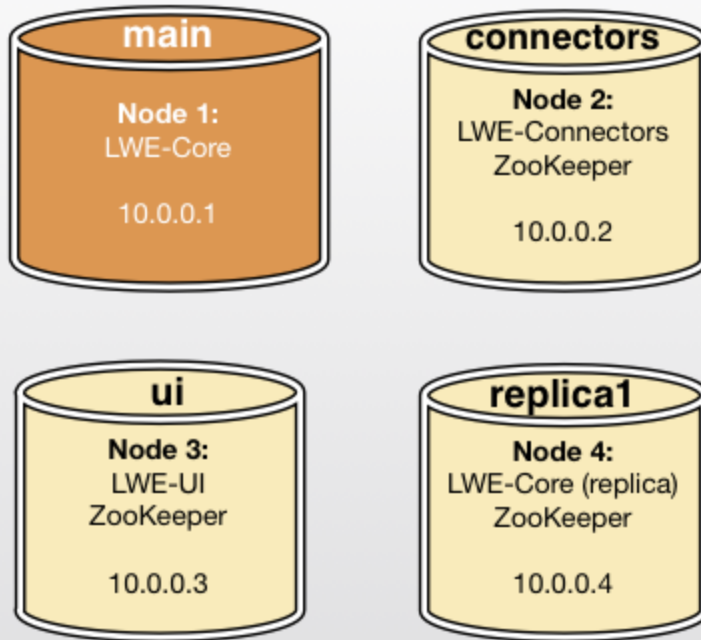
LucidWorks Search inherits all of Solr's SolrCloud functionality, so many of the recommendations for a SolrCloud cluster apply to LucidWorks Search. However, because LucidWorks Search adds [Admin UI and Connectors as components](#) in their own JVMs, there are some additional considerations when planning the cluster.

When planning your cluster, there are several factors to keep in mind:

- You need a quorum of ZooKeeper instances in an ensemble. This means there should be an odd number of ZooKeeper instances because in order for ZooKeeper to work properly, a majority of the total number of machines must be up. So, if you have two instances and one goes down, there would not be a majority up. If you have 3 instances, however, and 1 goes down, 2 are up so a majority are available. See the ZooKeeper documentation on [Clustered \(Multi-Server\) Setup](#) for more information.
- It is possible to install LucidWorks Search so each component runs on a different node of a cluster. However, at the present time, this requires a full installation of LucidWorks Search on each node, even if it will only run a single component. LucidWorks Search also needs to be installed on any node that will serve as a replica.
- While many of the Solr features are supported in SolrCloud mode, some of the LucidWorks Search features are not yet. If the three LucidWorks Search components (Core, Connectors and UI) are located on the same server, this won't be generally noticeable. However, if the components are on different servers, there must be one Core component that is the central point of communication for the Connectors and the UI.

A Sample Cluster

Sample Cluster of LucidWorks Search in SolrCloud mode



The graphic at right shows one possible configuration. Let's step through the nodes and discuss what they are used for.

The "main" Node

The node labeled "main" has the LWE-Core component (which includes Solr and the LucidWorks' API). This is called "main" because even though the LWE-Core component will be installed on node 3 also, the "main" node is the one that will be the point of communication for the LWE-UI and LWE-Connectors components. It is important to have the LWE-Connectors and LWE-UI components only work with one of the LWE-Core nodes (the "main" node) because not all of LucidWorks Search configuration is yet cloud-ready.

ZooKeeper

After the "main" node, the remaining three nodes host ZooKeeper in an ensemble configuration. ZooKeeper does not have to be co-located with LucidWorks Search, and LucidWorks Search does not need to be installed on with ZooKeeper; the ZooKeeper nodes can be located on any other nodes of a cluster. However, it should always be configured on an odd number of nodes. This is because the ensemble configuration relies on a quorum of ZooKeeper servers, meaning the majority of servers must be live for ZooKeeper to be able to continue serving requests if one node goes down. If there are only two nodes, one node down is half the available nodes, which is not a majority. Having an odd number of nodes gives a greater chance of surviving an outage to one or more nodes.

The "connectors" Node

This node hosts the LWE-Connectors component, which run each of the connectors available with LucidWorks Search. These connectors only connect to the remote repositories, crawl the content, and prepare Solr input documents for indexing. The documents are then passed to the LWE-Core component for indexing. It makes sense to install this component on its own node if you expect to do a lot of crawling while in production, or if you want to restrict outgoing network access to only one node of your cluster.

The "ui" Node

This node hosts the LWE-UI component, which contains the Admin UI and the Alerts functionality. Note that user queries from your search application do not get sent to this node. User queries are always direct to Solr, so they would go to the LWE-Core node.

Since the hardware requirements for the LWE-UI component are light, you could use the associated LWE-Core as a replica.

The "replica1" Node

This is again a full installation of LucidWorks Search, but is intended to be a replica for the main node. You can have as many of these as you would like for failover.

Related Topics

- [Cluster Installation](#)
- [Working With LucidWorks Search Components](#)

Cluster Installation

One of the most exciting features in Solr 4.x is SolrCloud, which allows simplified sharding and replication of Solr collections and indexes across multiple machines. LucidWorks Search includes Solr, and inherits this functionality.

The LucidWorks Search installer allows installation of Apache ZooKeeper and LucidWorks Search across multiple servers through the use of a configuration file that defines the role and location of each node of the cluster. This configuration file is then copied with the installer to each node and ZooKeeper or LucidWorks Search is installed as required. The benefit of this approach is that the installer is aware of the definitions for each other node, and can fill in the required address or port values as needed saving the system administrator some time during the actual installation.

This page contains several sections:

- [Using the Installer for a Cluster Installation](#)
- [Installer Configuration File](#)
 - [Profiles](#)
 - [Instances](#)
 - [ZooKeeper](#)
- [Installer Logs](#)
- [Installer Command Line Commands](#)
- [Example Cluster Installations](#)
 - [Simple Getting Started Cluster](#)
 - [Cluster with Four LucidWorks Search Instances: Two Shards](#)
 - [Cluster with Four LucidWorks Search Instances: Four Shards](#)
- [Related Topics](#)

Before starting the installation, it may be helpful to review the section [Planning a Search Cluster](#).

Using the Installer for a Cluster Installation

To install LucidWorks Search on more than one node, the simplest approach is this general process:

1. Create a configuration file with the definition of each node.
2. Copy the configuration file and installer .jar or .exe to each node.
3. Run the installer on each node with command line options referencing the configuration file and defined node profiles.

It's also possible to run the installer on each node with the graphical installer, but with a large number of nodes this may be difficult to keep track of each node during installation.

Installer Configuration File

The configuration file is an XML file that allows you to define the structure of the cluster.

There are several sections of the configuration file, many of which can be used as needed. To help you get started, there are three sample configuration files attached to this page, described in detail (with links to the files) in the [Example Cluster Installations](#) section below.

- ✔ The examples below are Linux-based examples. If you are using Windows, you may need to modify the examples for your specific OS, such as using a backward-slash instead of a forward-slash.

Profiles

The `profiles` section of the file contains parameters for the LucidWorks Search application. These parameters include port definition and installation path.

Each profile is defined with the `profile` parameter and is assigned an `id` (which can be any string) that will be used later to associate the profile with an `instance`. For each profile, the LWS component for the profile is defined by assigning it a `port`. A profile that includes the `core` component can also use additional parameters for SolrCloud mode, ZooKeeper and JVM heap sizing (see below).

```
<profiles>
  <profile path="/Applications/LucidWorks/LucidWorksSearch" id="main">
    <core port="8888" numShards="2" />
    <connectors port="8765" />
    <ui port="8989" />
  </profile>
  <profile path="/Applications/LucidWorks/LucidWorksSearch" id="shard1">
    <core port="8888" />
  </profile>
</profiles>
```

In the above example, we have defined two profiles. The first profile has an `id` of "main". One of the profiles must be called "main", and there can only be one "main" profile. This profile is used with SolrCloud mode installations, and defines the node that will be the point of contact for the Connectors and UI components if those are installed on different servers.

In this example, we have installed all of the LucidWorks components with the same profile. We could install them on different servers by including other profiles that only define the Connectors and/or UI components and the port they will run on.

In the "main" profile example above, note there is an additional parameter defined for the Core component, called `numShards`. This is equivalent to the `numShards` parameter used when bootstrapping Solr, and defines the number of shards that will make up the SolrCloud cluster. The number of shards must be determined in advance: any core nodes that come online after the defined number of shards will be considered replicas for the sharded installation.

Any profile that will enable the LWE-Core component (defined as 'core' in the port definitions of a profile) can also take several other parameters:

- `javaMaxPerm`: Set the maximum PermGen size for the JVM. If not defined, the default will be used (256Mb).
- `javaXms`: Set the initial heap size for the JVM. If not defined, the default will be used (512Mb).
- `javaXmx`: Set the maximum heap size for the JVM. If not defined, the default will be used (1024Mb).
- `rootDir`: The root directory in ZooKeeper, where configuration files will be stored. The specified directory will be created and configuration files copied to it during the bootstrap process. If not defined, a default of 'lws' will be used.

The second profile in the example above ("shard1") has only the Core component defined. We can use this profile for installation on several different servers (via the `instances` section of the configuration file) and each install will run only the Core component on port 8888.

Instances

The `instances` section contains parameters for the servers where LucidWorks Search will be installed. Each instance is defined with a `url`, an `id` (which must be an integer), and then associated with profiles that were defined in the `profiles` section.

```
<instances>
  <instance url="http://10.0.1.1" id="1">
    <profiles>main</profiles>
  </instance>
  <instance url="http://10.0.1.2" id="2">
    <profiles>shard1</profiles>
  </instance>
</instances>
```

In this example, we have two different instances defined and have associated them with profiles previously defined. The `url` is the hostname and protocol that will be used for other LucidWorks Search instances to communicate with other instances, no matter their role. So, the `url` should always be entered beginning with HTTP or HTTPS, and followed by the IP or hostname of the intended server.

As mentioned in the section on profiles, we can define multiple instances and re-use profiles. So, if we wanted to have 4 shards, we would change the `numShards` parameter to 4, then define four instances. Three of those instances would reference the same `profile` (profile "shard" in this example), and LucidWorks Search would be installed on the same path and use the same port on each machine. For example:

```
<instances>
  <instance url="http://10.0.1.1" id="1">
    <profiles>main</profiles>
  </instance>
  <instance url="http://10.0.1.2" id="2">
    <profiles>shard</profiles>
  </instance>
  <instance url="http://10.0.1.3" id="3">
    <profiles>shard</profiles>
  </instance>
  <instance url="http://10.0.1.4" id="4">
    <profiles>shard</profiles>
  </instance>
</instances>
```

Note that only one instance can use the "main" profile, but that instance can have other profiles defined for it as long as the `path` for the profile is different from the `path` in another profile that will be installed on the same machine.

ZooKeeper

The `zookeeper` section contains parameters for the ZooKeeper ensemble. It includes definition of `profiles` and `nodes`, which define the installation paths, ports, and hosts for each node that will host ZooKeeper.

The first step is to again define at least one `profile` for ZooKeeper, which has an `id` (must be an integer). The `profile` also includes the installation path, the data directory, ports, and other parameters. Then, `nodes` are defined and assigned one of the profiles.

```
<zookeeper>
  <profiles>
    <profile installPath="/usr/local/zookeeper/" id="1">
      <dataDir value="/usr/local/zookeeper/zkdata" />
      <port value="3001" />
      <electionPort value="4001" />
      <clientPort value="5001" />
      <tickTime value="2000" />
      <syncLimit value="5" />
      <initLimit value="10" />
    </profile>
  </profiles>
  <nodes>
    <node id="1" host="10.0.0.11" profile="1" />
    <node id="2" host="10.0.0.12" profile="1" />
    <node id="3" host="10.0.0.13" profile="1" />
  </nodes>
</zookeeper>
```

In many cases, you will likely have a single profile. However, you may want to set up multiple profiles so each ZooKeeper installation runs on different ports, or for other reasons. In the example above, we have a single profile, but have three nodes defined. That means we will install ZooKeeper on each node with the same definitions on each node.

The parameters defined are ZooKeeper configuration parameters. A short summary of the parameters is included here, but for full details, please see the [ZooKeeper Administrator's Guide](#).

Parameter	Description
installPath	The path on the server where ZooKeeper will be installed. For installation with the LucidWorks installer, this directory should not exist or be empty.
dataDir	The location where ZooKeeper will store database snapshots and transaction logs. For installation with the LucidWorks installer, this directory should not exist or be empty.
port	The ZooKeeper server port.
electionPort	The port that will be used for leader election.
clientPort	The port that will be used for the clients to communicate.
tickTime	The length of a "tick", which is the time unit used with ZooKeeper, measured in milliseconds. The default is 2000 milliseconds.
syncLimit	The amount of time, in ticks, to allow nodes to sync with a leader. If the node falls too far behind a leader, it will be dropped.
initLimit	The amount of time, in ticks, to wait for nodes to connect and sync to a leader.

The `nodes` section defines each node, with each one having an `id` (which must be an integer), a `host`, and an associated ZooKeeper `profile`. The `host` must be a hostname or an IP address, without any protocol (such as `tcp://` or `http://`) defined. The `profile` will be one of the profiles defined in the ZooKeeper section of the configuration file.

It is not possible, using this installation approach, to use the embedded ZooKeeper that is included with Solr. This approach will only install an externally-managed ZooKeeper, and it must always be an ensemble. If you would like to use the embedded ZooKeeper, install LucidWorks in standalone and bootstrap it manually with the instructions in the section [Using SolrCloud in LucidWorks](#).

Installer Logs

The installer will create a log file each time it is run named `installer.<date>.log` (the date is expressed in Epoch time). In many cases, the logs repeat what is shown on the screen, but if there are errors, they can be preserved for communication with [LucidWorks Support](#) as needed.

Once installation is complete and the system started with no errors, the log files can be deleted.

Installer Command Line Commands

Once the configuration file is prepared, it can be used with the installer by passing options at the command line to reference the defined instances, nodes and profiles.

The installer accepts the following parameters:

Option	Description	When to Use	Use with Parameters
-b	Starts LucidWorks search in SolrCloud mode (i.e., bootstraps the configuration and copies configuration files to ZooKeeper).	After installation of the instance using the "main" profile, when it will be run in SolrCloud mode.	-f, -instance, -profile
-f <i>configFile</i>	Defines the configuration file name. It may also include the path to the file if not in the same directory as the installer.	The configuration file should always be specified in every command line call.	All
-i	Installs LucidWorks Search without any bootstrap configuration (i.e., not in SolrCloud mode).	After a standalone installation, or when installing an instance that will not be a shard or replica in a SolrCloud cluster.	-f, -instance, -profile
-instance <i>instanceID</i>	Selects a specific instance from the configuration file (by ID).	During any type of LucidWorks Search installation.	-b, -f, -profile, -r, -s, -v
-nh	Disables the System Usage Monitor. By default, the System Usage Monitor will be enabled.	When you have elected to not send anonymous system data to LucidWorks.	-f, -i, -s
-node <i>nodeID</i>	Selects the ZooKeeper node to be installed.	During ZooKeeper installation.	-zki, -f
-profile <i>profileID</i>	Selects one of the profiles for an instance.	During any type of LucidWorks Search installation.	-b, -f, -instance, -r, -s, -v
-r	Starts the LucidWorks Search components. The required start scripts (<i>start.sh</i> or <i>start.bat</i>) and configuration files (<i>master.conf</i> are verified as being available.	To start any LucidWorks Search instance that does not need to bootstrap configuration files to ZooKeeper, or when restarting an instance that has already been bootstrapped.	-f, -instance, -profile

-s	Installs LucidWorks Search in SolrCloud mode. This will insert ZooKeeper parameters (zkHost) into the LucidWorks <code>master.conf</code> file so subsequent LWS starts correctly connect with ZooKeeper.	When you want the 'core' component to be a shard or replica of a SolrCloud cluster.	-f, -instance, -profile
-v	Verifies that the bootstrap process was successful.	When you want to check the bootstrap process worked.	-f, -instance, -profile
-w	Installs LucidWorks Search as a Windows Service (on Windows machines only).	When you want LucidWorks Search to be run as a service on a Windows machine.	
-zki	Installs ZooKeeper.	During any ZooKeeper installation.	-f, -node
-zkv	Verifies the ZooKeeper ensemble. This should only be invoked once all ZooKeeper nodes are installed and online.	When you want to verify a ZooKeeper ensemble is up, has a leader and each node is communicating with other nodes.	-f

Example Cluster Installations

In these examples, please note that your local path to the installation file and the name of the installer file may differ from what is used below. If using the sample configuration files, please note you will need to change paths, URL and host definitions for your environment before you start the installation process.

Simple Getting Started Cluster

The configuration file for this example is attached to this page as `TwoShardExample.xml`. If using this example, please modify it for your environment before starting.

This cluster could be used when first working with SolrCloud on distributed machines. It will install two LucidWorks Search instances on different servers and three ZooKeeper nodes on a single server. This may be helpful while setting up your cluster environment or doing testing, but it should not be used in production or for testing that is meant to mirror production performance or stability. The point of having a ZooKeeper ensemble is in case of hardware or other system failure; if all three nodes are on a single server, they will likely all go down when one goes down.

The LucidWorks instances will be installed on two different servers. One install will run all three LucidWorks Search components (Core, UI, and Connectors) from a single instance. The second install will run only the Core component, as a second SolrCloud shard.

The installer and configuration files need to be copied to each node or server where it will be run. Modify paths to the installer and name of installer file as needed in the examples below.

Step 1: Install ZooKeeper as three nodes on the same server

Node 1:

```
java -jar Downloads/lucidworks-search-installer.jar -zki -node 1 -f TwoShardExample.xml
```

Node 2:

```
java -jar Downloads/lucidworks-search-installer.jar -zki -node 2 -f TwoShardExample.xml
```

Node 3:

```
java -jar Downloads/lucidworks-search-installer.jar -zki -node 3 -f TwoShardExample.xml
```

Step 2: Start the Ensemble

To start the ensemble, start each node individually by moving to

`$ZK_install/zookeeper-3.4.5/bin` (where `$ZK_install` is the `installPath` in the configuration file).

On Linux-based servers:

```
./zkServer.sh start
```

On Windows-based servers, you omit defining 'start' in the command:

```
./zkServer.sh
```

Repeat this step for every node of the ensemble.

Step 3: Verify ZooKeeper Installation

This step ensures that each node is aware of other nodes and they can communicate.

```
java -jar Downloads/lucidworks-search-installer.jar -zkv -f TwoShardExample.xml
```

Step 4: Install First LucidWorks Instance ("main")

```
java -jar Downloads/lucidworks-search-installer.jar -s -instance 1 -profile main -f TwoShardExample.xml
```

Step 5: Bootstrap First LucidWorks Instance

```
java -jar Downloads/lucidworks-search-installer.jar -b -instance 1 -profile main -f TwoShardExample.xml
```

Step 6: Install Second LucidWorks Instance ("shard")

```
java -jar Downloads/lucidworks-search-installer.jar -s -instance 2 -profile shard1 -f
TwoShardExample.xml
```

Step 7: Start Second LucidWorks Instance

```
java -jar Downloads/lucidworks-search-installer.jar -r -instance 2 -profile shard1 -f
TwoShardExample.xml
```

Cluster with Four LucidWorks Search Instances: Two Shards

The configuration file for this example is attached to this page as FourInstanceExample.xml. If using this example, please modify it for your environment before starting.

In this example, ZooKeeper is installed on three servers, as a true ensemble. Four LucidWorks instances will be installed, on four different servers. One instance will run all three LucidWorks Search components (Core, UI, and Connectors) from a single instance. The second install will run only the Core component, as a second SolrCloud shard. The three LucidWorks Search components will be split across three instances: one instance will run the Core component, one the UI component and a third the Connectors component. The fourth instance will run only the Core component, as a second SolrCloud shard. Note in the sample configuration file that the `numShards` value has been set to '2', since only two of the four installations are meant to be shards.

The installer and configuration files need to be copied to each node or server where it will be run. Modify paths to the installer and name of installer file as needed in the examples below.

Step 1: Install ZooKeeper on each node of the ensemble

Node 1:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 1 -f
FourInstanceExample.xml
```

Node 2:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 2 -f
FourInstanceExample.xml
```

Node 3:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 3 -f
FourInstanceExample.xml
```

Step 2: Start the Ensemble

To start the ensemble, start each node individually by moving to

`$ZK_install/zookeeper-3.4.5/bin` (where `$ZK_install` is the `installPath` in the configuration file).

```
./zkServer.sh start
```

Repeat this step for every node of the ensemble.

Step 3: Verify ZooKeeper Installation

This step ensures that each node is aware of other nodes and they can communicate.

```
java -jar /Downloads/lucidworks-search-installer.jar -zkv -f FourInstanceExample.xml
```

Step 4: Install First LucidWorks Instance ("main")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 1 -profile main -f FourInstanceExample.xml
```

Step 5: Bootstrap First LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -b -instance 1 -profile main -f FourInstanceExample.xml
```

Step 6: Install Second LucidWorks Instance ("connectors")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 2 -profile connectors -f FourInstanceExample.xml
```

Step 7: Start Second LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 2 -profile connectors -f FourInstanceExample.xml
```

Step 8: Install Third LucidWorks Instance ("ui")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 3 -profile ui -f FourInstanceExample.xml
```

Step 9: Start Third LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 3 -profile ui -f FourInstanceExample.xml
```

Step 10: Install Fourth LucidWorks Instance ("shard1")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 4 -profile shard1 -f
FourInstanceExample.xml
```

Step 11: Start Fourth LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 4 -profile shard1 -f
FourInstanceExample.xml
```

Cluster with Four LucidWorks Search Instances: Four Shards

The configuration file for this example is attached to this page as FourShardExample.xml. If using this example, please modify it for your environment before starting.

In this example, ZooKeeper is installed on three servers, as a true ensemble. Four LucidWorks instances will be installed, on four different servers. One instance will run all three LucidWorks Search components (Core, UI, and Connectors) from a single instance. The second, third, and fourth instances will run only the Core component as SolrCloud shards. Note in the sample configuration file that the numShards value has been set to '4'.

The installer and configuration files need to be copied to each node or server where it will be run. Modify paths to the installer and name of installer file as needed in the examples below.

Step 1: Install ZooKeeper on each node of the ensemble

Node 1:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 1 -f
FourShardExample.xml
```

Node 2:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 2 -f
FourShardExample.xml
```

Node 3:

```
java -jar /Downloads/lucidworks-search-installer.jar -zki -node 3 -f
FourShardExample.xml
```

Step 2: Start the Ensemble

To start the ensemble, start each node individually by moving to \$ZK_install/zookeeper-3.4.5/bin (where \$ZK_install is the installPath in the configuration file).

```
./zkServer.sh start
```

Repeat this step for every node of the ensemble.

Step 3: Verify ZooKeeper Installation

This step ensures that each node is aware of other nodes and they can communicate.

```
java -jar /Downloads/lucidworks-search-installer.jar -zkv -f FourShardExample.xml
```

Step 4: Install First LucidWorks Instance ("main")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 1 -profile main -f  
FourShardExample.xml
```

Step 5: Bootstrap First LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -b -instance 1 -profile main -f  
FourShardExample.xml
```

Step 6: Install Second LucidWorks Instance ("shard")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 2 -profile shard -f  
FourShardExample.xml
```

Step 7: Start Second LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 2 -profile shard -f  
FourShardExample.xml
```

Step 8: Install Third LucidWorks Instance ("shard")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 3 -profile shard -f  
FourShardExample.xml
```

Step 9: Start Third LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 3 -profile shard -f  
FourShardExample.xml
```

Step 10: Install Fourth LucidWorks Instance ("shard")

```
java -jar /Downloads/lucidworks-search-installer.jar -s -instance 4 -profile shard -f  
FourShardExample.xml
```

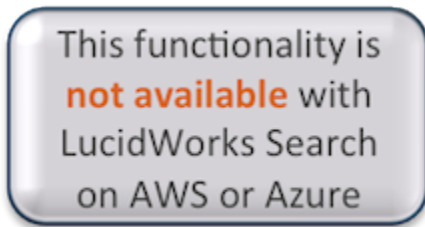
Step 11: Start Fourth LucidWorks Instance

```
java -jar /Downloads/lucidworks-search-installer.jar -r -instance 4 -profile shard -f  
FourShardExample.xml
```

Related Topics

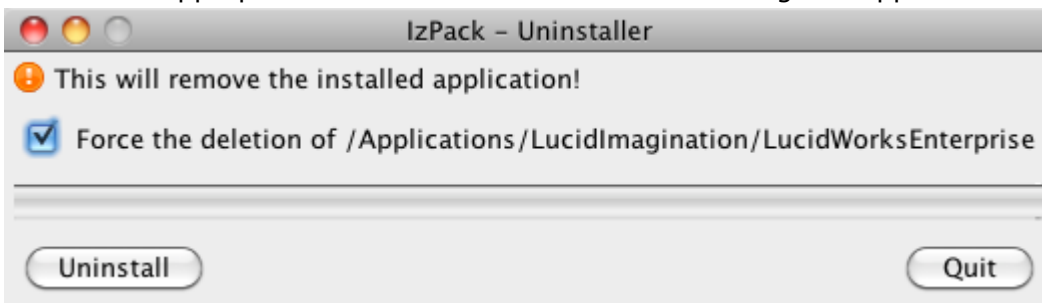
- [Starting and Stopping LucidWorks Search](#)
- [Working With LucidWorks Search Components](#)

Uninstalling LucidWorks



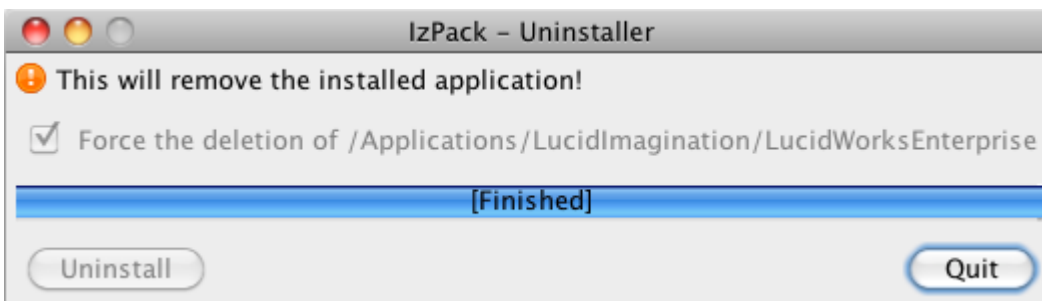
You can uninstall LucidWorks Search by running the uninstaller found in the `$LWS_HOME/app/uninstaller` directory. Two files are available: `uninstaller.exe` for Windows systems and `uninstaller.jar` for Linux and Mac systems.

1. Launch the appropriate file. The IzPack - Uninstaller dialog box appears:



2. Select **Force the deletion of your/installation/directory** to remove the parent installation directory. If you do not force the deletion of the installation directory, the application will be removed but the installation directory will remain.
3. Click **Uninstall**.

The uninstaller displays the progress bar.



4. When the uninstallation is complete, click **Quit** to close the uninstaller.

**Uninstall from the Command Line**

You cannot run the uninstaller from the command line in console mode. To remove LucidWorks Search on a server without GUI access, [stop all running LucidWorks processes](#), then manually delete the parent directory. This will remove all indexes and associated data.

Upgrading from a Prior Version

This functionality is
not available with
LucidWorks Search
on AWS or Azure

The upgrade process for LucidWorks Search includes several steps to update the system configuration files, the search indexes, the LucidWorks software and any embedded software from third-party vendors (such as Solr and Lucene, etc.).

It's recommended that the process described below be performed on a test system before upgrading your production application.

- ❗ If you are moving to LucidWorks Search v2.6.3 from a version lower than 2.6.2, there are additional configuration changes required to support changes in the UI applications introduced with v2.6. These changes are detailed in Step 6 below, and please take care to complete them before starting LucidWorks Search after the upgrade process.

Supported Versions

The upgrade process described in this section only applies to upgrades within v2.x. If you are using v1.7 or v1.8 of LucidWorks, and would like to move to v2.6, you will first need to move to v2.0. Contact [LucidWorks Support](#) for help with an upgrade from v1.x to v2.x.

If you are using v2.0.x, you can upgrade directly to v2.6.x.

Customers who are using any version 2.5.2 or higher can skip the step to run the Index Upgrade Tool. That step is only required for LucidWorks Search versions before 2.5.x. The reason it is required is because the older versions included a pre-release snapshot of the Solr 4.x branch; LucidWorks Search versions after 2.5.x include only officially released versions of Solr 4.x.

Upgrade Steps

Upgrading LucidWorks requires the following steps:

- [Preparation: Review the Upgrade Notes](#)
- [Step 1: Stop Existing Installation and Create a Backup](#)
- [Step 2: Install the New Version of LucidWorks](#)
- [Step 3: Run the MetadataUpgradeTool](#)
- [Step 4: Upgrade the Indexes or Delete Them \(upgrades from 2.1.1 or earlier ONLY\)](#)
- [Step 5: Update the \\$LWS_HOME/app dir](#)
- [Step 6: Make Other Manual Changes](#)
- [Step 7: Start LucidWorks](#)

Preparation: Review the Upgrade Notes

The following instructions contain step-by-step information about the upgrade process. There may, however, be specific changes for each release and for your specific implementation that need to be considered before, during, or after the upgrade. Before starting any upgrade, please review the list of changes for your version in the section **Upgrade Notes**.

Also note any manual changes you may have made to configuration files found in `$LWS_HOME/conf/master.conf`. You will need to re-apply those changes manually in [Step 6: Make Manual Changes to Configuration Files](#), described below.

Step 1: Stop Existing Installation and Create a Backup

LucidWorks Search can be stopped with the scripts available in `$LWS_HOME/app/bin`. Use either `stop.sh` or `stop.bat` depending on your operating system.

A backup of your existing installation can be made by copying the entire `$LWS_HOME` directory to a parallel location. If your installation has been made in a directory called `/usr/local/LucidWorks`, then in Unix, for example, the command would be:

```
cp -r /usr/local/LucidWorks /tmp/lws-old-backup
```

You can name the backup whatever you'd like, as long as you remember it later if you need it.

Now is also a good time to make note of any customizations that will be over-ridden by the upgrade, if you haven't done so already. For details of these changes, see the Upgrade Notes for your version.

[Back to Top](#)

Step 2: Install the New Version of LucidWorks

Follow the steps outlined in the [sections on installing LucidWorks](#) to install the new version of LucidWorks. This can be done in any location on your server, but you should change the default path if your existing LucidWorks is already installed there. You can choose the same ports as are used in your existing installation: this installation will only be used to get an updated `app` directory, the `MetadataUpgradeTool`, and the `IndexUpgradeTool` (see [Step 3](#) and [Step 4](#) below).

If you are using LucidWorks Search in SolrCloud mode, you would perform the installation on each node that is running LucidWorks Search. Each of the following steps would also need to be run on each node, as appropriate for the LucidWorks Search version (details in each step).



Do not install the new version of LucidWorks Search over your existing installation!

When the installer asks if you would like to start the new version of LucidWorks, say no (uncheck the box). If you are using the installer to update your SolrCloud cluster, do not run the steps to bootstrap or start any LucidWorks Search instances.

The instructions from here on will refer to the top level of this installation as `$LWS_NEW`.

[Back to Top](#)

Step 3: Run the MetadataUpgradeTool

The MetadataUpgradeTool is found in the `$LWS_NEW/app/migration_tools` directory. It will update your configuration files to the proper format for the new version and will also move the user database from the `app` dir to the `data` dir if it still exists in the old location.

To run it, issue this command:

```
$ java -jar $LWS_NEW/app/migration_tools/MetadataUpgradeTool.jar -verbose -lwe_conf_dir  
<existing conf dir> -lwe_app_dir <existing app dir> -lwe_data_dir <existing data dir>
```


The parameters `-lwe_conf_dir`, `-lwe_app_dir`, and `-lwe_data_dir` are required and should point to the `$LWS_HOME/conf`, `$LWS_HOME/app` and `$LWS_HOME/data` directories, respectively, of your existing LucidWorks application.

The parameter `-verbose` is optional, but highly recommended. It allows the full output of the tool to show on-screen in case there are problems or failures. If support is needed after using the MetadataUpgradeTool, Lucid Imagination will request this output. The output will look something like:

```
### Upgrading format of collections.yml ...
Writing upgraded collections.yml
### Upgrading similarity configuration in Solr schema ...
Upgrading similarity configuration in
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/collection1_0/conf/schema.
upgraded
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/collection1_0/conf/schema.
similarity configuration in
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/LucidWorksLogs/conf/schema
upgraded
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/LucidWorksLogs/conf/schema
Upgrading field mapping configuration in solrconfig ...
Processing
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/collection1_0/conf/solrcon
...
Writing upgraded
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/collection1_0/conf/solrcon
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/LucidWorksLogs/conf/solrco
...
Writing upgraded
/Applications/LucidImagination/lwe2.0.1-upgrade/conf/solr/cores/LucidWorksLogs/conf/solrco
Removing old autocomplete data files...
### Upgrading UI DB location...
Moved
/Applications/LucidImagination/lwe2.0.1-upgrade/app/webapps/lwe-ui/WEB-INF/db/production.s
/Applications/LucidImagination/lwe2.0.1-upgrade/data/lwe-ui
### Upgrading UI DB...
Executing UI DB upgrade
== AddExternalProtocolToSettings: migrating =====
-- change_table(:settings)
  -> 0.0060s
== AddExternalProtocolToSettings: migrated (0.0070s) =====
```



If you have created any collection templates they **will not** be upgraded with the MetadataUpgradeTool. It's recommended that all templates created with previous versions be recreated from scratch using a new or upgraded collection.

 One of the changes from v2.0 to v2.1 was to change the names of the data source types "S3" and "S3N". The "S3N" type, for Amazon over S3, is now known as "S3". The former "S3" type, for a Hadoop Filesystem on S3, is now known as "S3H". During the upgrade process, these types will be converted automatically.

However, there is a new requirement for these types of data sources that the value for the URL must include a trailing slash if the URL points to a directory of files and not to a single resource. This is not done automatically by the upgrade. In order to successfully crawl an existing "S3" or "S3H" data source after upgrading, the trailing slash must be added manually if the path specified is not to a single file or resource.

[Back to Top](#)

Step 4: Upgrade the Indexes or Delete Them (upgrades from 2.1.1 or earlier ONLY)


If you are upgrading from v2.5.2 or later to v2.6, **do not** run this step. Skip to [Step 5: Update the \\$LWS_HOME/app dir](#). There are no index changes that need to be made with this tool, you can use them as is.

If you are using v2.1.1 or earlier, you **must** complete this step.

The MetadataUpgradeTool only upgrades configuration and other system files. However, the search indexes also need to be upgraded as a separate step. If, however, the indexed content is not needed, the indexes can be simply deleted instead of upgraded.

Upgrading the Indexes

In order to upgrade an index from an earlier version of LucidWorks, the Index Upgrade Tool should be run. This tool is found under `$LWS_HOME/app/migration-tools/` in a .jar file called `IndexUpgradeTool.jar`.

 It is recommended to run this Upgrade Tool only after consultation with [Lucid Imagination Support](#) to be sure you understand the full ramifications of running this tool on your local, customized, index.

While we have provided this index upgrade tool to help you avoid re-indexing your data, it is recommended to do so with every migration to a new version.

Before running the tool, you should make sure LucidWorks is not running to ensure there are no indexing processes taking place while performing the upgrade.

To run the tool, open a command line interface, navigate to the `$LWS_NEW/app/migration_tools` directory and issue the command:

```
$ java -jar IndexUpgradeTool.jar [options] <sourcedir> <destinationdir>
```

The `<sourcedir>` parameter is the existing index that will be upgraded and moved to the `<destinationdir>`. Unlike the `MetadataUpgradeTool`, which updates configuration files in place, the `IndexUpgradeTool` makes a copy of the index while upgrading it. The `<sourcedir>` is `$LWS_HOME/data/solr/cores/collection/data/index` (where *collection* is the name of the collection to be upgraded). The `destinationdir` should be a temporary location (preferably a directory, such as `/tmp/index-upgrade`, as the output is a number of raw index files).

- ✔ The `IndexUpgradeTool` can upgrade the index for one collection at a time. If there are multiple collection in the origin LucidWorks installation, these would each need to be converted separately. By default, there are at least two indexes: the one that contains your data (called *collection1* at initial installation) and one that indexes log data called `LucidWorksLogs`. Don't forget to upgrade the `LucidWorksLogs` collection or delete the index - skipping that collection will cause LucidWorks to fail on restart.

There is currently one option that can be used with the `IndexUpgradeTool`, which is `-checkindex`. This will run Lucene's `checkindex` tool to validate that the index is correct. This is a good idea to do, especially for systems already in production, but will add time to the upgrade process.

The output of the tool is a number of index files in the location you specified with the `<destinationdir>` parameter. Once the tool has finished, delete the existing index files for each collection (found under `$LWS_HOME/data/solr/cores/collection/data/index`) and copy the new index files for each collection to the `index` dir.

Deleting the Indexes

If the index files are not particularly needed, which may be the case for the `LucidWorksLogs` collection particularly, they can be deleted. To do so, delete all of the subdirectories found under `$LWS_HOME/data/solr/cores/collection/` for each collection.

i If you are using the DirectSpellChecker, which is the LucidWorks Search default spell check implementation, your spell check data will be recreated automatically with either index upgrade method described above (i.e., either by re-indexing all content or upgrading the index with the IndexUpgradeTool). This is because the DirectSpellChecker uses the main index to create spelling suggestions.

If, however, you have changed the default implementation and are using the IndexBasedSpellChecker, your spell check data is not incorporated with the main index, but instead stored as a separate index. In this case, these indexes will not be upgraded with the IndexUpgradeTool (nor deleted by deleting the main index) and may cause problems when restarting LucidWorks. If possible, these indexes should be deleted and rebuilt after the main index has been upgraded.

[Back to Top](#)

Step 5: Update the \$LWS_HOME/app dir

Neither the MetadataUpgradeTool nor the IndexUpgradeTool update the program files that make up the LucidWorks application. These need to be updated manually by replacing the `app` directory in the original installation with the `app` directory from the new LucidWorks Search installation. To do this, delete the original `$LWS_HOME/app` and copy the `$LWS_NEW/app` directory to `$LWS_HOME`. Assuming default locations of the installations, the sequence would be:

```
$ rm -r /LucidWorks/LucidWorksSearch/app
$ cp -r /LucidWorks/LWS-newVersion/app /LucidWorks/LucidWorksSearch
```

[Back to Top](#)

Step 6: Make Other Manual Changes

Update Jetty Context Configurations

In v2.6.2, we made some small modifications to the way the UI applications are bundled with LucidWorks Search. To support these changes, the context configurations must be updated with the proper path to the applications. Without these changes, the Relevancy Workbench, Quick Start, Flare, and Launchpad will not be available upon restart (the main Admin UI will be available, however - it's context has not changed in 2.6.2). If you are upgrading from 2.6.2 to 2.6.3, you do not need to make these changes. However, if you are upgrading from 2.6.0, then you should make these changes.

The details of the configuration files to be changed and the changes required is below:

- In `$LWS_HOME/conf/jetty/lwe-ui/contexts/flare.xml`
 - *Old:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/lw_flare.war</Set>`
 - *New:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/flare</Set>`
- In `$LWS_HOME/conf/jetty/lwe-ui/contexts/launchpad.xml`
 - *Old:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/launchpad.war</Set>`
 - *New:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/launchpad</Set>`
- In `$LWS_HOME/conf/jetty/lwe-ui/contexts/quickstart.xml`
 - *Old:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/quickstart.war</Set>`
 - *New:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/quickstart</Set>`
- In `$LWS_HOME/conf/jetty/lwe-ui/contexts/relevancy.xml`
 - *Old:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/relevancy.war</Set>`
 - *New:* `<Set name="war"><SystemProperty name="lucidworksAppHome"/>/webapps/relevancy</Set>`

Update master.conf

The upgrade process does not modify the `master.conf` file found in `$LWS_HOME/conf/`. Two changes are recommended for optimal performance in typical situations. They are not added automatically by the `MetadataUpgradeTool` because these or similar changes may have already been made in your implementation depending on your specific situation. Both parameters are entered in the section of the file for *JVM Settings for LWE-UI*:

- Add `-XX:MaxPermSize=256M`
- Add `-Dcom.sun.management.jmxremote`

Depending on the 2.x version you started with, these parameters may already be correct.

If you have added any other changes to `master.conf`, such as adding the `zkHost` for SolrCloud startup or performance-related changes, you should note those changes in the original `master.conf` file and re-apply those to the new `master.conf`.

Update solr.xml in SolrCloud installations

In SolrCloud installations, you should update your `solr.xml` file to include a system parameter for the `genericCoreNodeNames` in the `<cores>` section of the file. The `<cores>` section should look something like this:

```
<cores adminPath="/admin/cores" host="${host:}" hostPort="${jetty.port:}"
defaultCoreName="collection1" genericCoreNodeNames="${genericCoreNodeNames:true}">
```

If you do not add this parameter, it's possible that you will not be able to create collections with the upgraded instance.

Remove Crawl Data

If you have removed the indexes instead of upgrading them, and are using the Web, Aperture-based File System or Windows Share crawlers, you may also want to remove the crawl history for any data sources of that type. Otherwise, the crawl statistics that are shown may be inaccurate and documents may seem to be "missing" from the index. The reason for this is that the upgrade process does not clear the crawl history, but if the documents are removed because the index was deleted, the crawler may skip documents because it believes they have been seen and are unchanged.

To clear the crawl history, use the Data Source Crawl Data Delete API with a command such as:

```
curl -X DELETE
http://localhost:8888/api/collections/collection1/datasources/3/crawldata
```

You should replace the "collection1" with the name of the collection you are working with, and replace "3" with the real ID of the data source.

Other Changes

Other manual changes may be required depending on the version and your own implementation. Review the Upgrade Notes for your version for details.

[Back to Top](#)

Step 7: Start LucidWorks

Start LucidWorks Search using the scripts available in `$LWS_HOME/app/bin`. See the section [Starting and Stopping LucidWorks Search](#) for details on how to start LucidWorks Search.



After the upgrade, the indexes that drive auto-complete will have been removed. The auto-complete indexes are not automatically generated unless there is already an auto-complete activity scheduled (and it will happen on its schedule). If using auto-complete in your search application, and it is not scheduled to run periodically, use the Activity API or the [Admin UI](#) to schedule it and recreate the auto-complete indexes.

After checking to be sure the upgrade was successful, the `$LWS_NEW` installation and/or the backup can be deleted. In order to be sure the upgrade was successful, try to index some content and perform some queries, while checking for errors in the "core" log found in `$LWS_HOME/data/logs`.

[Back to Top](#)

Related Topics

- [Upgrade Notes](#)
- [Known Issues](#)

Migrating from Solr to LucidWorks

It is possible to move an existing Solr 3.x implementation to LucidWorks to use some of the LucidWorks features. Many of the LucidWorks features are configured in standard Solr configuration files, and LucidWorks is generally able to add those configuration details to configuration files as they are enabled during implementation of LucidWorks. Examples of this are Click Scoring, which has several components that will be added programmatically when it is properly enabled.

❌ Migrating from any Solr release prior to v3.x (such as 1.4.1) is **not possible** at this time. To move to LucidWorks from Solr 1.4.1 or earlier, you must first migrate to Solr 3.x. The technique to do that is out of scope for this document, but some assistance may be found in a LucidWorks blog post by Chris Hostetter on upgrading his [Solr 1.4 instance to 3.1](#).

In addition, these instructions assume you have a good deal of experience using Solr's configurations, and that you understand the concepts of RequestHandlers, UpdateHandlers and update chains.

The following items need to be added to the `schema.xml` and `solrconfig.xml` files for each collection:

- [Add the /lucid RequestHandler](#)
- [Add the lucid-update-chain Update Request Processor Chain](#)
- [Add /update RequestHandler](#)
- [Add Required Fields](#)
- [Working with Existing Solr Indexes](#)



Information for LucidWorks Search in the Cloud Users

Customers using Solr who would like to migrate to LucidWorks Search on AWS or Azure need only contact [LucidWorks Support](#) for assistance in making the move.

An existing Solr `schema.xml` and `solrconfig.xml` file should be used as the starting point, but LucidWorks-specific update handlers, request handlers and update chains must be added for a seamless experience. Once these items are added, LucidWorks features that require configuration in `solrconfig.xml` or `schema.xml`, such as Click Scoring, de-duplication, distributed updates (pre-SolrCloud), search filtering, field mapping with built-in crawlers, and some query parser parameters, will be added to those configuration files automatically when they are enabled.

However, configuration definitions for the auto-complete and spell check components are not added. If they are not already implemented in your Solr configuration, and you do not add them during this process (or at any point by manually editing `solrconfig.xml`), you **will not be able to enable them using the LucidWorks UI or API**. The components can be added manually at any time; once they are added, LucidWorks can be used to manage them (disable them). If they are not added manually, the Admin UI will warn users who try to use it that the components are not defined, so the feature cannot be enabled.



If it's expected that several collections in LucidWorks will be created based on an existing Solr configuration, it may be useful to modify the configuration of a single collection (such as the default collection1), then use that collection to create a template that will be used for other collections. For more on Collection Templates, see the section Using Collection Templates.

Add the /lucid RequestHandler

The `/lucid` RequestHandler is referenced with several of the features of LucidWorks, and must be added to `solrconfig.xml` for each collection. It is a Solr search request handler that is manageable by LucidWorks.

```
<requestHandler class="solr.StandardRequestHandler" name="/lucid" default="true">
  <lst name="defaults">
    <str name="defType">dismax</str>
    <str name="q.alt">*:*</str>
    <str name="qf">title^5.0 body</str>
    <str name="defOp">AND</str>
    <str name="fl">id,title,body,data_source_type,data_source_name,data_source</str>
  </lst>
</requestHandler>
```

Add the lucid-update-chain Update Request Processor Chain

LucidWorks updates the update chain if you enable some functionality. For example, a signature processor is added if you enable de-duplication. Similarly, a field mapping processor is added if you use the crawlers included with LucidWorks instead of Solr-specific document indexing techniques.

The following must be added to `solrconfig.xml` for each collection in order for these features to be properly enabled.

```
<updateRequestProcessorChain name="lucid-update-chain">
  <processor class="solr.LogUpdateProcessorFactory">
    <int name="maxNumToLog">10</int>
  </processor>
  <processor class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

Add /update RequestHandler

The /update RequestHandler is a standard Solr update request handler and should be configured to use the lucid-update-chain. Add the following to the `solrconfig.xml` for each collection:

```
<requestHandler class="solr.XmlUpdateRequestHandler" name="/update">
  <lst name="defaults">
    <str name="update.chain">lucid-update-chain</str>
  </lst>
</requestHandler>
```

Add Required Fields

LucidWorks indexing and searching depends on several fields being configured in the `schema.xml` for each collection. Add the following:

```

<types>
  <fieldType name="string" class="solr.StrField"/>
  <fieldType name="date" class="solr.TrieDateField" omitNorms="true"
precisionStep="0" positionIncrementGap="0"/>
  <fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
    <analyzer type="index">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter catenateAll="0" catenateNumbers="1" catenateWords="1"
class="solr.WordDelimiterFilterFactory" generateNumberParts="1"
      generateWordParts="1" splitOnCaseChange="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ASCIIFoldingFilterFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter catenateAll="0" catenateNumbers="0" catenateWords="0"
class="solr.WordDelimiterFilterFactory" generateNumberParts="1"
      generateWordParts="1" splitOnCaseChange="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ASCIIFoldingFilterFactory"/>
    </analyzer>
  </fieldType>
</types>
<fields>
  <dynamicField name="attr_*" type="string" indexed="true" stored="true"
multiValued="true" />

  <field indexed="true" multiValued="false" name="id" omitNorms="true" stored="true"
type="string"/>
  <field indexed="true" multiValued="false" name="data_source_type" omitNorms="true"
stored="true" type="string" omitTermFreqAndPositions="true"/>
  <field indexed="true" multiValued="false" name="data_source_name" omitNorms="true"
stored="true" type="string" omitTermFreqAndPositions="true"/>
  <field indexed="true" multiValued="false" name="data_source" omitNorms="true"
stored="true" type="string"/>
  <field default="NOW" indexed="true" multiValued="false" name="timestamp"
omitNorms="true" stored="true" type="date"/>
  <field indexed="true" multiValued="true" name="text_all" omitNorms="false"
stored="false" type="text_en"/>
  <field indexed="true" multiValued="false" name="batch_id" omitNorms="true"
stored="true" type="string"/>
  <field indexed="true" multiValued="true" name="title" omitNorms="false"
stored="true" type="text_en"/>
  <field indexed="true" multiValued="true" name="body" omitNorms="false"
stored="true" type="text_en"/>
</fields>
<uniqueKey>id</uniqueKey>
<defaultSearchField>text_all</defaultSearchField>

```


Working with Existing Solr Indexes

The indexes in your Solr instance will not work with LucidWorks automatically. There have been changes to the index codecs which require deciding if you will reindex your content from scratch or if you will upgrade the indexes and continue to work with your existing content upload strategy.

To figure out which option works best for you, it helps to understand a little bit about how LucidWorks stores documents and deals with data sources.

First, LucidWorks is using Solr 4, also known as trunk, which contains significant changes to index format. At a minimum, you will need to upgrade your indexes.

In addition, LucidWorks includes several embedded crawlers which are designed to acquire content and add it to the index. These crawlers are configured through the use of data sources, which include information about individual content repositories and how to access them. All of the content added to the index via data sources includes information about what data source added the document to the index, which is used for statistics and other features of the Admin UI. The content added to your existing Solr indexes will not have any of this information, unless you happened to add it, but then it likely will not be stored in fields which the LucidWorks Admin UI is able to understand (for example, the ID of a data source is stored in the `data_source` field, which is unlikely to exist in your existing content or index).

Re-indexing All Documents

This is the recommended approach because it is the least prone to problems going forward. This approach requires deleting your entire Solr indexes and re-indexing the content. Re-indexing the content can be done with the LucidWorks Admin UI or Data Sources API, or with using the same mechanisms used to index content in Solr. If choosing the same mechanisms as used with Solr, LucidWorks provides the option to create "External" data sources, which allow pushing content with your code directly to Solr in a way that links the incoming data with the External data source so the Admin UI and APIs can be fully utilized as though the embedded crawlers discovered the content and indexed it. For more information on External Crawlers, please review the section on Suggestions for External Data Sources.

Upgrading the Indexes

Upgrading the indexes is an alternative option, but is more prone to risk and may limit your use of all the LucidWorks features. LucidWorks includes an IndexUpgradeTool, which updates the format of the indexes. The IndexUpgradeTool **only works on Solr 3.x or higher indexes**. After using this tool, you can continue to use LucidWorks as a shell around Solr (accessing Solr directly as you are accustomed), but statistics about the content of your indexes will not be fully integrated with the LucidWorks Admin UI. This option may be a consideration for those who have a great deal of content in their indexes and for whom re-indexing will take a long period of time or is otherwise not an option.

This tool is found under `$LWS_HOME/app/migration-tools/` in a .jar file called `IndexUpgradeTool.jar`.

- ❗ It is recommended to run this Upgrade Tool only after consultation with LucidWorks Support to be sure you understand the full ramifications of running this tool on your local index.

While we have provided this index upgrade tool to help you avoid re-indexing your data, it is recommended to do so with every migration to a new version.

Before running the tool, you should make sure LucidWorks Search is **not running** and ensure there are no indexing processes taking place while performing the upgrade.

To run the tool, open a command line interface, navigate to the `$LWS_NEW/app/migration_tools` directory and issue the command:

```
$ java -jar IndexUpgradeTool.jar [options] <sourcedir> <destinationdir>
```

The `<sourcedir>` parameter is the existing index that will be upgraded and moved to the `<destinationdir>`. Unlike the `MetadataUpgradeTool`, which updates configuration files in place, the `IndexUpgradeTool` makes a copy of the index while upgrading it. The `<sourcedir>` is the location of the current indexes to be upgraded. The `destinationdir` should be a temporary location (preferably a directory, such as `/tmp/index-upgrade`), as the output is a number of raw index files.

- ✅ The `IndexUpgradeTool` can upgrade the index for one Solr core at a time. If there are multiple cores in the origin Solr, these would each need to be converted separately.

There is currently one option that can be used with the tool, which is `-checkindex`. This will run Lucene's `check index` tool to validate that the index is correct. This is a good idea to do, especially for systems already in production, but will add time to the upgrade process.

The output of the tool is a number of index files in the location you specified with the `<destinationdir>` parameter. Once the tool has finished, copy the new index files to the appropriate location for each upgraded core (LucidWorks cores are called collections, and are found under `$LWS_HOME/data/solr/cores/collection/data/index`).

Upgrade Notes for v2.6

A major feature of this release is the Relevancy Workbench, a tool to allow administrators to test possible configuration changes to improve or modify relevance ranking using their real content, before making the changes permanently.

Please review the Known Issues for v2.6 for additional late-breaking issues you should be aware of.

- [Before Upgrading](#)
- [System Changes](#)
- [Data Source Changes](#)
- [API Changes](#)
- [UI Changes](#)
- [Solr Update](#)

Before Upgrading

1. The installer has significantly changed and now supports installation of several LucidWorks Search instances in a SolrCloud cluster.
2. This upgrade includes an upgrade to Solr 4.4. While we try to highlight changes we expect will effect our customers, please note that any customizations you have made may behave differently with the new version of Solr. Please consult the [Solr release notes for 4.4](#) if you have any concerns about customizations you have made.
3. The default setting for `enablePositionIncrements` in `schema.xml` has changed in Lucene to "true" and disabling them has been deprecated (and will be removed entirely in a future version).
4. In general, the parsing of the `schema.xml` has become stricter in Solr 4.3 and 4.4, which means that typos, deprecated settings or other errors in `schema.xml` may cause startup failures. The solution is to remove the deprecated or incorrect entries and start again; some changes may additionally require a system restart.
5. When upgrading from any 2.5.x version to v2.6, it is not required to run the Index Upgrade Tool to upgrade your indexes, and you can continue with the indexes you have in place without re-indexing your content. If you are migrating from any version prior to v2.5.x (such as 2.1.1 or 2.0, etc.), you must run the Index Upgrade Tool as your index format will not be compatible with v2.5.x or v2.6.
6. As with all upgrades, any configuration files contained inside a collection template will not be upgraded. If you have created collection templates, they will need to be re-created based on an upgraded collection or modified manually as defined in these notes. Configuration files inside existing collections that were made with a template, however, will be upgraded (meaning, the fact that a collection was made with a template does not impact its ability to be upgraded).

[Back to Top](#)

System Changes

- A new installer has been included to allow cluster installation of LucidWorks Search in SolrCloud mode with a command-line installer and a configuration file.

Related Documentation

- SolrCloud Cluster Installation
[Back to Top](#)

Data Source Changes

- A new option for crawl output has been added. Now, JavaScript can be run on the crawl output to transform the crawl output as needed.
- The MapR crawler has been improved and some parameters have changed; a second MapR crawler for high-volume crawling has also been added. To use either of these data sources, you must have a MapR client installed on the same server as the LWE-Connectors component.
- A new parameter called "Max retries" has been added to a Web data source to control the number of times LucidWorks Search will try to crawl the data source before removing documents from the index.
- Several new parameters have been added to the Windows Shares data source to provide better Active Directory and LDAP support.

Related Documentation

- Data Sources
- MapR Data Sources and MapR High Volume Data Sources in the REST API
- [MapR Data Source](#) and [MapR High Volume Data Sources](#) in the Admin UI
- Windows Shares Data Sources REST API or [Windows Share Data Source](#) in the Admin UI
[Back to Top](#)

API Changes

- The Collections API now uses Solr's Collection API, which allows setting the `replicationFactor` and `max_shards_per_node` parameters when in SolrCloud mode.

Related Documentation

- Collections
[Back to Top](#)

UI Changes

- The admin UI was previously available at 'http://localhost:8989' (depending on the servername and port used for the LWE-UI component). That default URL (called the Launch Pad) will now show a choice of options, with three new UI features:
 - Quick Start, a wizard-like interface for quickly configuring a Web, File, or JDBC data source and seeing search results as soon as possible.
 - Relevancy Workbench, which allows administrators to see the impact of possible parameter changes on result sets before making the changes permanent in `solrconfig.xml` or their search applications.
 - LucidWorks Flare, an integration with Blacklight to provide a production-ready search UI that's easily customizable.
- The main LucidWorks Search Admin UI and a link directly to the Solr Admin are also now available from the Launch Pad.
- There are now two options for local filesystem crawls. The second option has always been available via the API, but now it is exposed in the UI also.
- The UI has been further redesigned for improved usability. The default Search UI is now available under a Tools, where the link to the Solr Admin UI has also been moved.

Related Documentation

- [Quick Start](#)
- [Relevancy Workbench](#)
- [LucidWorks Flare](#)

Solr Update

The embedded Solr in LucidWorks Search has been updated to include Solr v4.4. We have also incorporated the following patches:

For more information about the integration of LucidWorks and Solr, see the section [Solr Direct Access](#).

[Back to Top](#)

Working With LucidWorks Search Components

LucidWorks Search has three main components that can each be run together on a single server or deployed on separate servers if desired. While LucidWorks Search customers on AWS or Azure will not often need to interact with these components, an understanding of how they work is helpful for a deeper understanding of the system as a whole.

- [About the Components](#)
 - [LWE-Core](#)
 - [LWE-UI](#)
 - [LWE-Connectors](#)
 - [Default Installation URLs](#)
- [Configuring the Components](#)
- [Related Topics](#)

About the Components

Each component is a single JVM process. The system properties for each JVM can be modified with the `master.conf` file found in the `$LWS_HOME/conf` directory.

LWE-Core

The LucidWorks Search Core component is the main engine of the application. It contains the search index, the index definitions, the query parser, the embedded Solr application and Lucene libraries, as well as serves the REST API (with the exception of Alerts).

LWE-UI

The UI component includes all web-based graphical interfaces for administering the application, a sample search interface, Relevancy Workbench and the enterprise alerts feature.

Through the Admin UI, you can modify index fields, configure data sources for content collection, define aspects of the search experience, and monitor system performance.

The Search UI provides a front-end for users to submit queries to LucidWorks Search and review results. It is not intended as a production-grade user interface, rather as a sample interface to use while configuring and testing the system.

Relevancy Workbench is a tool to model possible changes to how user query terms are interpreted in order to improve relevancy. More information about this tool is available at [Relevance Workbench](#).

Enterprise Alerts provide a way for users of the front-end Search UI to save searches and receive notifications when new results match their query terms. There is a user interface piece with forms and screens for users to configure and review their alerts, as well as a REST API for programmatic access to the Alerts features.

LWE-Connectors

The Connectors component performs all the crawler functions, which include crawling data sources on demand or at a specific schedule, maintaining a crawl history (as applicable; each crawler varies in their behavior), and saving data source configuration information for use by the crawlers. The Connectors component also manages the [LucidWorks Logs](#) crawler.

Default Installation URLs

This guide will refer to example URLs that will reference the default installation URLs for each component. These defaults are:

Component	Default URL	Web Interfaces
LWE-Core	http://127.0.0.1:8888/	This URL is used as the base for accessing most of the REST APIs, and also for accessing Solr Admin UI at http://127.0.0.1:8888/solr . .
LWE-UI	http://127.0.0.1:8989/	There are multiple front-ends at this URL. This base URL will access the Landing Page, which will provide access to the Quick Start, the LucidWorks Search Admin UI, Relevancy Workbench, and also a link to the Solr Admin UI.
LWE-Connectors	http://127.0.0.1:8765/	There is no web front-end at this URL, it is used by the LWE-Core and LWE-UI components to communicate with the Connectors component.

These URLs are used by the installer for two purposes:

1. When the various components communicate with each other, or link to one another, they specify which URL will be used.
2. If the "Enable" check box is selected for a component when using the installer, then that component will be run locally, using the port specified in the URL.



The default LucidWorks start scripts start all components at the same time. However, it is possible to restart or stop a single component. See the section [Starting and Stopping LucidWorks Search](#) for details.

[Back to Top](#)

Configuring the Components

If all components are run on the same machine, they must be defined with different ports. They can also be configured to run on different servers.

There are three possible ways to configure the components:

1. All components run on the same machine and they are started and stopped together. This is the default for the [standalone installer](#), which automatically prompts for default ports that are different for each component. To use this mode, you only need to run the installer once and follow through the process completely.
2. All components run on the same machine but they are started and stopped separately. This would require running the installer three times on the same machine. See [Installing Components on Different Servers](#) for detailed instructions on how to do this.
3. Each component is on a different machine and started and stopped separately. This requires running the installer on each machine. See [Installing Components on Different Servers](#) below for detailed instructions on how to do this.

[Back to Top](#)

Related Topics

- [Expanding Capacity](#)

System Directories and Logs

This functionality is
not available with
LucidWorks Search
on AWS or Azure

There are several important directories in the LucidWorks Search installation. System activities are recorded in several log files. Knowing where files and logs are located will make system configuration and troubleshooting easier.

- [Locating Files and Directories](#)
 - [Configuring LucidWorks Search Directories](#)
 - [Temporary Files](#)
- [System Logs](#)
 - [Log Properties](#)
- [LucidWorksLogs Collection](#)
- [Related Topics](#)

Locating Files and Directories

The following table shows the default location of some directories that may be needed to effectively work with LucidWorks Search. These paths are all relative to the LucidWorks Search installation path (referred to as `$LWS_HOME`) which is specified [during installation](#).

What	Path
Configuration Files	<code>\$LWS_HOME/conf/</code>
Documentation	<code>\$LWS_HOME/app/docs/</code> (PDF) or http://docs.lucidworks.com (Online)
Examples	<code>\$LWS_HOME/app/examples/</code>
Jetty Libraries	<code>\$LWS_HOME/app/jetty/lib/</code>
Licenses	<code>\$LWS_HOME/app/legal/</code>
Logs	<code>\$LWS_HOME/data/logs/</code> (See below for log file list)
LucidWorks Indexes	<code>\$LWS_HOME/data/solr/cores/collection/data/</code>
LucidWorks Logs	<code>\$LWS_HOME/data/solr/cores/LucidWorksLogs/data/</code>
Solr Home	<code>\$LWS_HOME/conf/solr/</code>
Solr Configuration Files	<code>\$LWS_HOME/conf/solr/cores/collection/conf/</code>

Solr Source Code	\$LWS_HOME/app/solr-src/
Start/Stop Scripts	\$LWS_HOME/app/bin/



Editing Configuration Files on Windows

LucidWorks Search holds configuration files open after reading them, which may cause problems on Windows systems that do not allow editing open files. In this case, stop LucidWorks Search before editing files on Windows to be sure the edits are saved properly.

Configuring LucidWorks Search Directories

After you have installed LucidWorks Search, you can configure the location of the `app`, `conf`, `data`, and `logs` directories by passing these parameters to the start script (`start.sh` or `start.bat`):

- `-lwe_app_dir`
- `-lwe_conf_dir`
- `-lwe_data_dir`
- `-lwe_log_dir`

For example, to change the location of the `data` directory, pass the following parameter to your start script:

```
start.sh -lwe_data_dir /var/data
```

See the section on [Starting and Stopping LucidWorks Search](#) for more information about the start scripts.

Temporary Files

By default, LucidWorks Search uses standard system directories (as detected by the JVM) for creating temporary files. This can be changed by adding a system property to the `master.conf` for `java.io.tmpdir` in the section that controls each JVM for the system. For example, to change the location of temporary files for the LucidWorks Core component, you would follow these steps:

1. Shut down LucidWorks using the instructions found in the section on [Starting and Stopping LucidWorks Search](#).
2. Open `master.conf` with a text editor (found in `$LWS_HOME/conf`).
3. Find the section for `lwecore.jvm.params` and add `-Djava.io.tmpdir=/tmp/files/`.
4. Start LucidWorks.

The directory chosen as the location for temporary files should exist before starting LucidWorks Search, and must be writable by the user running LucidWorks.

[Back to Top](#)

System Logs

LucidWorks Search records system activities to rolling log files located in the `$LWS_HOME/data/logs` directory of the installation by default. The table below describes the main purpose of the various log files.

Log Name	Name Pattern	Function
Connector component log	<code>connectors.<YYYY_MM_DD>.log</code>	Connectors component operations, including the output of all crawling operations.
Connector request log	<code>connectors.request.<YYYY_MM_DD>.log</code>	Requests to the connectors component. These usually come from the Core component.
Core component log	<code>core.<YYYY_MM_DD>.log</code>	LucidWorks Core component operations, such as indexing.
Core request log	<code>core.request.<YYYY_MM_DD>.log</code>	Requests to the core component. These could come from either the Connectors or the UI component.
Core standard error log	<code>core-stderr.log</code>	Errors from Jetty startup (if any).
Core standard output log	<code>core-stdout.log</code>	Messages from Jetty startup (if any).
UI component log	<code>ui.<YYYY_MM_DD>.log</code>	Information from the Rails application, which runs the Search, Admin and Alerts components.
UI request log	<code>ui.request.<YYYY_MM_DD>.log</code>	Requests to the UI component.
Ruby standard error log	<code>ruby-stderr.log</code>	Errors from Ruby startup (if any).
Ruby standard output log	<code>ruby-stdout.log</code>	Messages from Ruby startup (if any).
Click log	<code>click-<collectionName>.log</code>	User click data, for use in relevance boosting (if enabled).

SharePoint crawl log	google_connectors.feed.log	SharePoint crawling operations. Note, this file can also include a number in the name, such as google_connectors.feed0.log, etc.
-------------------------	----------------------------	---

Log files are available through the Admin UI, by going to the Server Logs page for a collection and clicking the link at the bottom of the page. If for some reason the Admin UI is not available, log files can be downloaded with a curl command to the Core component such as:

```
curl http://localhost:8888/logs/<log_file_name>
```

Note, however, if the LucidWorks Search Core component is down, that curl command will not work.

Log Properties

The LucidWorks Search Core log is configured by the `$LWS_HOME/conf/log4j-core.xml` properties file. The default is to create a distinct log per date (server time).

The LucidWorks Search UI log is configured by the `$LWS_HOME/conf/log4j-ui.xml` properties file. The default is to create a distinct log per date (server time).

The LucidWorks Search Connector log is configured by the `$LWS_HOME/conf/log4j-connectors.xml` properties file. The default is to create a distinct log per date (server time).



The LucidWorks Search Connectors log includes information about crawl activities such as attempts to access a file or URL and the results of those attempts. By default, the log does not record the collection or data source associated with crawl activities. However, if you would like to record that information for later review, you can edit the `$LWS_HOME/conf/log4j-connectors.xml` file.

In the file, find the section that begins with a comment to "Use the pattern below to log additional context info...", as below:

```
<!-- Use the pattern below to log additional context info like collection and
data source name -->
<!--
  <param value="%d{ISO8601} %p %c{2} - %X %m%n" name="ConversionPattern"/>
-->
```

Uncomment `<param value="%d{ISO8601} %p %c{2} - %X %m%n" name="ConversionPattern"/>` and save the file. You should restart LucidWorks Search after making this change.

More information on how to modify log4j settings for the Core and UI log files is available at <http://logging.apache.org/log4j/1.2/manual.html>.

[Back to Top](#)

LucidWorksLogs Collection

LucidWorks Search records log files for your Solr indexes in a collection called LucidWorksLogs, which contains a pre-configured data source also called `lucidworkslogs`. You can view the data for the LucidWorksLogs collection as you would for any other collection. You can also access the log files directly in the `$LWS_HOME/data/solr/cores/LucidWorksLogs/` directory.

The LucidWorksLogs collection powers the error log and all statistics about recent query and indexing activity that is shown in the Admin UI.

The log files on a LWE-Core server are accessible via HTTP at the URL `"http://server:port/logs"`. This URL lists all files currently in the logs directory, and provides links for downloading them individually. This can be useful in situations where you do not have direct shell access to the LWE-Core machine, but would like to review the log files for troubleshooting purposes.

If you are using LucidWorks Search in SolrCloud mode or with each component installed on a different server, please see the section Log Indexing with Separated Components for details on how to make sure your logs are fully indexed.

When securing the HTTP Port of LWE-Core installation, consideration should be taken as to whether the `"/logs"` directory should be secured or not.



Deleting the LucidWorksLogs Collection

It is possible to delete the LucidWorksLogs collection if desired; however, this will disable the server log page within other collections, all activity graphing, and all calculations of Most Popular and Most Recent queries.

If the collection was deleted in error, or if you'd like to restore it at a later time, go to the Server log page within any collection and click **Recreate the log collection**.

It is also possible to remove the LucidWorksLogs data source from the LucidWorksLogs collection (i.e., retain the collection for possible later use, but remove the mechanism that indexes the logs). However, at the current time it will automatically be re-created and re-scheduled on server restart. If you wish to disable log crawling, you must either remove the entire LucidWorksLogs collection, or modify the LucidWorksLogs data source so that the schedule is not active (you can modify the schedule with the Data Source Schedules API or in the Schedules screen of the Admin UI).

Related Topics

- [Working With LucidWorks Search Components](#)
- [Starting and Stopping LucidWorks Search](#)

[Back to Top](#)

Starting and Stopping LucidWorks Search

This functionality is
not available with
LucidWorks Search
on AWS or Azure

LucidWorks Search can be started and stopped using start and stop scripts provided with the application. These scripts are described below.

- ✔ Windows users who have configured LucidWorks Search to run as a service should use the Services panel in Windows to manage start and stop.

- [Starting a Standalone LucidWorks Search Instance](#)
- [Starting SolrCloud-enabled LucidWorks Search Instances](#)
 - [Passing SolrCloud parameters at Start](#)
 - [Updating `master.conf`](#)
- [Stopping LucidWorks Search \(all modes\)](#)
- [Starting or Stopping Components Separately](#)

Starting a Standalone LucidWorks Search Instance

If you did not allow the installer to start LucidWorks Search, or if shortcuts were not installed, you can still start or stop the system manually from the command line. This will start all components:

1. Open a command shell, and make sure Java 1.6 or greater is in your path.
2. Change directories to the LucidWorks installation directory, then to the `$LWS_HOME/app/bin` directory.
3. Invoke `start.sh` for UNIX/Mac/Cygwin or `start.bat` for Windows systems.

- ⚠ If you are using LucidWorks Search in SolrCloud mode, please refer to the section [Starting LucidWorks Search in the documentation for Using SolrCloud in LucidWorks Search](#).

Starting SolrCloud-enabled LucidWorks Search Instances

If you are using LucidWorks Search in SolrCloud mode, you must start the application in a way that the underlying Solr instances are aware of where ZooKeeper is. If you used the LucidWorks Search installer, the required parameters have been added to the `conf/master.conf` file for each instance.

However, if you bootstrapped LucidWorks Search manually, or installed without the all of the SolrCloud installer steps, you will need to pass the required parameters on the command line. You can also manually update `conf/master.conf` file.

Passing SolrCloud parameters at Start

As long as the initial bootstrap has been completed (if not, please see Starting LucidWorks Search), the only parameter that is required on future startup is the `zkHost` parameter. This parameter points to each of the ZooKeeper instances and the root directory for the configurations that are stored in ZooKeeper. This example command starts LucidWorks Search and points to an external ZooKeeper:

```
$ ./start.sh -lwe_core_java-opts  
"-DzkHost=10.0.1.7:5001,10.0.1.9:5001,10.0.1.11:5001/lws"
```

If you are using the embedded ZooKeeper instance, then you may alternately need to start ZooKeeper while starting LucidWorks Search with the `zkRun` parameter on one of the instances. Subsequent instances would require the `zkHost` parameter to point to the instance with the running ZooKeeper. For example, to start the first instance:

```
$ ./start.sh -lwe_core_java-opts "-DzkRun"
```

Then all subsequent instances are started:

```
$ ./start.sh -lwe_core_java-opts "-DzkHost:10.0.1.7:9988"
```

Note when using the embedded ZooKeeper that the port is the LWE-Core component port + 1000.

Updating master.conf

If you don't want to have to pass the ZooKeeper parameters each time you restart, you can modify the `conf/master.conf` file for each instance. Simply add the `-DzkHost` parameters to the section JVM Settings of LWE-Core and they'll be passed to the start script. For example, here is a sample where:


```
# COMPONENT LWE-Core - LWE-Solr + LWE REST API.
# -----
lwecore.enabled=true
lwecore.address=http://10.0.1.5:8888

# JVM Settings for LWE-Core
lwecore.jvm.params=-Xms512M -Xmx1024M -XX:MaxPermSize=256M -Duser.language=en
-Duser.country=US -Duser.timezone=UTC -Dfile.encoding=UTF-8
-Dcom.sun.management.jmxremote -DzkHost=10.0.1.7:5001,10.0.1.9:5001,10.0.1.11:5001/lws
```

If using the embedded ZooKeeper instance, the same approach can be taken to add the `-DzkRun` parameter to one instance, with `-DzkHost` being added to the other instances.

These parameters only need to be added to the LWE-Core component for each instance that runs the LWE-Core component; so if you have an instance that is only running the UI or the Connectors, the parameters don't need to be added at all.

Stopping LucidWorks Search (all modes)

To stop LucidWorks Search, use the stop scripts at the command line. This will stop all components and any running processes.

1. Open a command shell, and make sure Java 1.6 or greater is in your path.
2. Change directories to the LucidWorks installation directory, then to the `$LWS_HOME/app/bin` directory.
3. Invoke `stop.sh` for UNIX/Mac/Cygwin or `stop.bat` for Windows systems.



Restarting LucidWorks Search

To restart LucidWorks Search, first stop the servers and start them again using the processes outlined above.

Starting or Stopping Components Separately

To start or stop any of the components without starting or stopping the other components, you can use the `start.sh/start.bat` or `stop.sh/stop.bat` scripts with the `-only` parameter, followed by the component name.

- Core component: `lwe-core`
- UI component: `lwe-ui`
- Connectors component: `connectors`

For example, this would start only the connectors using the `start.sh` script:

```
start.sh -only connectors
```

Glossary of Terms

Where possible, terms are linked to relevant parts of the documentation for more information.

Jump to a letter:

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

A

Alerts

An alert allows a user to save searches. There are two types: *active*, which will send notifications when new results are found, and *passive*, which do not send notifications.

Auto-Complete

A way to provide users suggestions for possible matching queries before they have finished typing. In LucidWorks Search, this relies on an index of terms to be created on a regular basis by scheduling it as an activity.

B

Boolean Operators

These control the inclusion or exclusion of keywords in a query by using operators such as AND, OR, and NOT.

C

Click Scoring Relevance Framework

A method of changing the relevance ranking of a document based on the number of times other users have clicked on the same document.

Collection

One or more documents grouped together for the purposes of searching. See also [Document](#).

Component

A part of LucidWorks Search that has been designed to stand alone or can be run independently from other components. LucidWorks Search has three main components: *LWE-Core*, which runs Solr, indexing, and other critical application functions, *LWE-Connectors*, which handles all crawling activities, and *LWE-UI*, which runs the Administrative UI, the front-end search interface, and the alerting functionality.

Connector

A connector is a program or piece of code that allows a connection to be made to a data source and content to be extracted from it.

Crawler

Also known as a "spider", this is a program that is able to retrieve documents internal or external servers.

D ---

Data Source

Defines the metadata required to connect to a location containing content to be indexed. It could be a file system path, a Web URL, a JDBC connection, or some other set of values.

Distributed Index

A distributed index is one where the search index for a [collection](#) is spread across more than one [shard](#).

Distributed Search

Distributed search is one where queries are processed across more than one [shard](#).

Document

One or more Fields. See also [Field](#).

F ---

Field

The content to be indexed/searched along with metadata defining how the content should be processed by LucidWorks Search.

I

Inverse Document Frequency (IDF)

A measure of the general importance of a term. It is calculated as the number of total Documents divided by the number of Documents that a particular word occurs in the collection. See <http://en.wikipedia.org/wiki/Tf-idf> and http://lucene.apache.org/core/old_versioned_docs/versions/3_5_0/scoring.html for more info on TF-IDF based scoring and Lucene scoring in particular. See also [Term Frequency](#).

Inverted Index

A way of creating a searchable index that lists every word and the documents that contain those words, similar to an index in the back of a book which lists words and the pages on which they can be found. When performing keyword searches, this method is considered more efficient than the alternative, which would be to create a list of documents paired with every word used in each document. Since users search using terms they expect to be in documents, finding the term before the document saves processing resources and time.

M

Metadata

Literally, *data about data*. Metadata is information about a document, such as it's title, author, or location.

N

Natural Language Query

A search that is entered as a user would normally speak or write, as in, "What is aspirin?"

Q

Query Parser

A query parser processes the terms entered by a user.

R

Recall

The ability of a search engine to retrieve *all* of the possible matches to a user's query.

Relevance

The appropriateness of a document to the search conducted by the user.

Replication

A method of copying a master index from one server to one or more "slave" or "child" servers. In LucidWorks Search, the master continues to manage updates to the index, while queries are handled by the slaves. This approach enables LucidWorks Search to properly manage query load and ensure responsiveness.

REST API

An alternative way of controlling LucidWorks Search without accessing the user interface.

S

Shard

A method of partitioning a database or search engine to maximize performance and efficiency.

SolrCloud

[Ongoing work](#) within the Solr community to improve Solr's ability to operate in a cloud environment.

Solr Schema (schema.xml)

The Apache Solr index schema. The schema defines the fields to be indexed and the type for the field (text, integers, etc.) The schema is stored in schema.xml and is located in the Solr home conf directory.

Solr Config (solrconfig.xml)

The Apache Solr configuration file. Defines indexing options, RequestHandlers, highlighting, spellchecking and various other configurations. The file, solrconfig.xml is located in the Solr home conf directory.

Spell Check

The ability to suggest alternative spellings of search terms to a user, as a check against spelling errors causing few or zero results. In LucidWorks Search, when spell-checking is enabled, a parallel "spell" index is created as documents are indexed.

Stopwords

Generally, words that have little meaning to a user's search but which may have been entered as part of a [natural language](#) query. Stopwords are generally very small pronouns, conjunctions and prepositions (such as, "the", "with", or "and")

Synonyms

Synonyms generally are terms which are near to each other in meaning and may substitute for one another. In a search engine implementation, synonyms may be abbreviations as well as words, or terms that are not consistently hyphenated. Examples of synonyms in this context would be "Inc." and "Incorporated" or "iPod" and "i-pod".

T

Term Frequency

The number of times a word occurs in a given document. See <http://en.wikipedia.org/wiki/Tf-idf> and http://lucene.apache.org/java/2_3_2/scoring.html for more info on TF-IDF based scoring and Lucene scoring in particular.

See also [Inverse Document Frequency \(IDF\)](#).

W

Wildcard

A wildcard allows a substitution of one or more letters of a word to account for possible variations in spelling or tenses. In LucidWorks Search, there are two ways to use them. One is to use an asterisk (*) at the end of a term to find all documents that contain words that start with that pattern. For example, `paint*` would find `paint`, `painter` and `painting`. A second way is to use a question mark (?) in the middle of a term to substitute for one character in that term. Such as, `c?t` would find `cat`, `cot` and `cut`. It's also possible to use wildcards at the start of a term in the same way - either to replace a single letter (using the ? symbol) or to find documents that contain words that end with a pattern using a *. For example, `*sphere` would find `ecosphere` and `stratosphere`.

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