Configuring Elasticsearch

Liferay DXP is an open source project, so you won't be surprised to learn that its default search engine is also an open source project. Elasticsearch is a highly scalable, full-text search and analytics engine.

By default, Elasticsearch runs as an embedded search engine, but it's only supported in production as a separate server or cluster. This guide walks you through the process of configuring Elasticsearch.

If you'd rather use Solr, it's also supported. See here for information on installing and configuring Solr.

To get up and running quickly with Elasticsearch as a remote server, refer to the Installing Elasticsearch article. In that article you'll find the basic instructions for the installation and configuration of Elasticsearch in a single server environment. This article includes more details and information on clustering and tuning Elasticsearch. In this article you'll learn to configure your existing Elasticsearch installation for use in production environments.

If you've come here looking for information on search engines in general, or the low level search infrastructure of Liferay DXP, refer instead to the developer tutorial Introduction to Liferay Search.

These terms will be useful to understand as you read this guide:

- *Elasticsearch Home* refers to the root folder of your unzipped Elasticsearch installation (for example, `elasticsearch-2.4.0`).
- *Liferay Home* refers to the root folder of your Liferay DXP installation. It contains the `osgi`, `deploy`, `data`, and `license` folders, among others.

Upgrading to Elasticsearch 6.5: If you have an existing 2.4 or 6.1 Elasticsearch installation, follow the upgrade guide to move onto Elasticsearch 6.5.

Embedded vs. Remote Operation Mode

both servers in the same process has drawbacks:

- Elasticsearch must use the same JVM options as Liferay DXP.
- Liferay DXP and Elasticsearch compete for resources.

You wouldn't run an embedded database like HSQL in production, and you shouldn't run Elasticsearch in embedded mode in production either. Instead, run Elasticsearch in *remote operation mode*, as a standalone server or cluster of server nodes.

Configuring Elasticsearch

For detailed Elasticsearch configuration information, refer to the Elasticsearch documentation.

The name of your Elasticsearch cluster is important. When you're running Elasticsearch in remote mode, the cluster name is used by Liferay DXP to recognize the Elasticsearch cluster. To learn about setting the Elasticsearch cluster name on the Liferay DXP side, refer below to the section called Configuring the Liferay Elasticsearch adapter.

Elasticsearch's configuration files are written in YAML and kept in the `[Elasticsearch Home]/config` folder:

- `elasticsearch.yml` is for configuring Elasticsearch modules
- `logging.yml` is for configuring Elasticsearch logging

To set the name of the Elasticsearch cluster, open `[Elasticsearch Home]/config/elasticsearch.yml` and specify

```
cluster.name: LiferayElasticsearchCluster
```

Since `LiferayElasticsearchCluster` is the default name given to the cluster in Liferay DXP, this would work just fine. Of course, you can name your cluster whatever you'd like (we humbly submit the recommendation `clustery_mcclusterface`).[1] You can configure your node name using the same syntax (setting the `node.name` property).

If you'd rather work from the command line than in the configuration file, navigate to Elasticsearch Home and enter

```
./bin/elasticsearch --cluster.name clustery_mcclusterface --node.name nody_mcnodeface
```

Starting Elasticsearch

Start Elasticsearch by navigating to Elasticsearch Home and typing

```
./bin/elasticsearch
```

if you run Linux, or

```
\bin\elasticsearch.bat
```

if you run Windows.

To run as a daemon in the background, add the `-d` switch to either command:

```
./bin/elasticsearch -d
```

When you have Elasticsearch itself installed and running, and Liferay DXP installed and running (do that if you haven't already) you need to introduce Liferay DXP and Elasticsearch to each other. Fortunately, Liferay provides an adapter that helps it find and integrate your Elasticsearch cluster.

Configuring the Liferay Elasticsearch Adapter

The Elasticsearch connector is a module that ships with the Foundation Suite and deployed to the OSGi runtime, titled *Liferay Portal Search Elasticsearch*. This connector provides integration between Elasticsearch and Liferay DXP. Before you configure the adapter, make sure Elasticsearch is running.

Elasticsearch 6.5: The connector for Elasticsearch 6.5 is called *Liferay Connector to Elasticsearch 6*. Download the Elasticsearch 6.5 connector from Liferay Marketplace and install it, following the Marketplace documentation if necessary.

There are two ways to configure the adapter:

1. Use the System Settings application in the Control Panel.

2. Manually create an OSGi configuration file.

need a configuration file so you can use the same configuration on another Liferay DXP system, you can still use System Settings. Just make the configuration edits you need, then export the `.config` file with your configuration.

CONFIGURING THE ADAPTER IN THE CONTROL PANEL

Here are the steps to configure the Elasticsearch adapter from the System Settings application:

1. Start Liferay DXP.

2. Navigate to *Control Panel* → *Configuration* → *System Settings* → *Foundation*.

3. Find the *Elasticsearch* entry (scroll down and browse to it or use the search box) and click the Actions icon (⋮), then *Edit*.

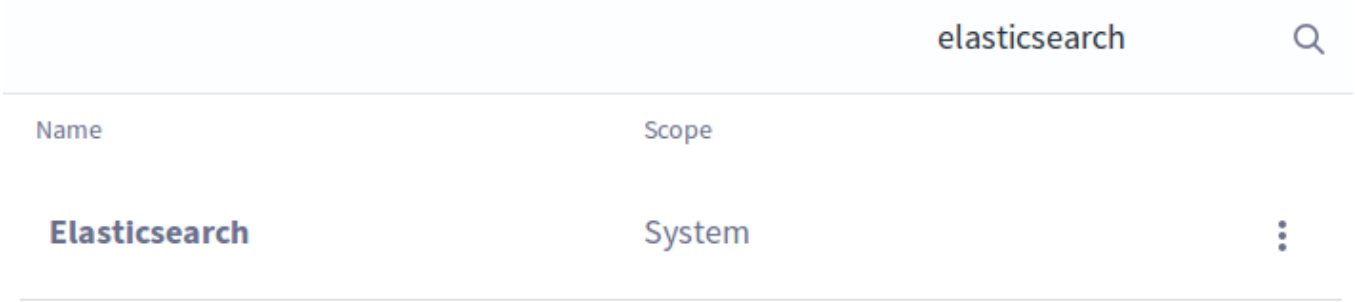


Figure 1: Use the System Settings application in Liferay DXP's Control Panel to configure the Elasticsearch adapter.

4. Change Operation Mode to *Remote*, and then click *Save*.

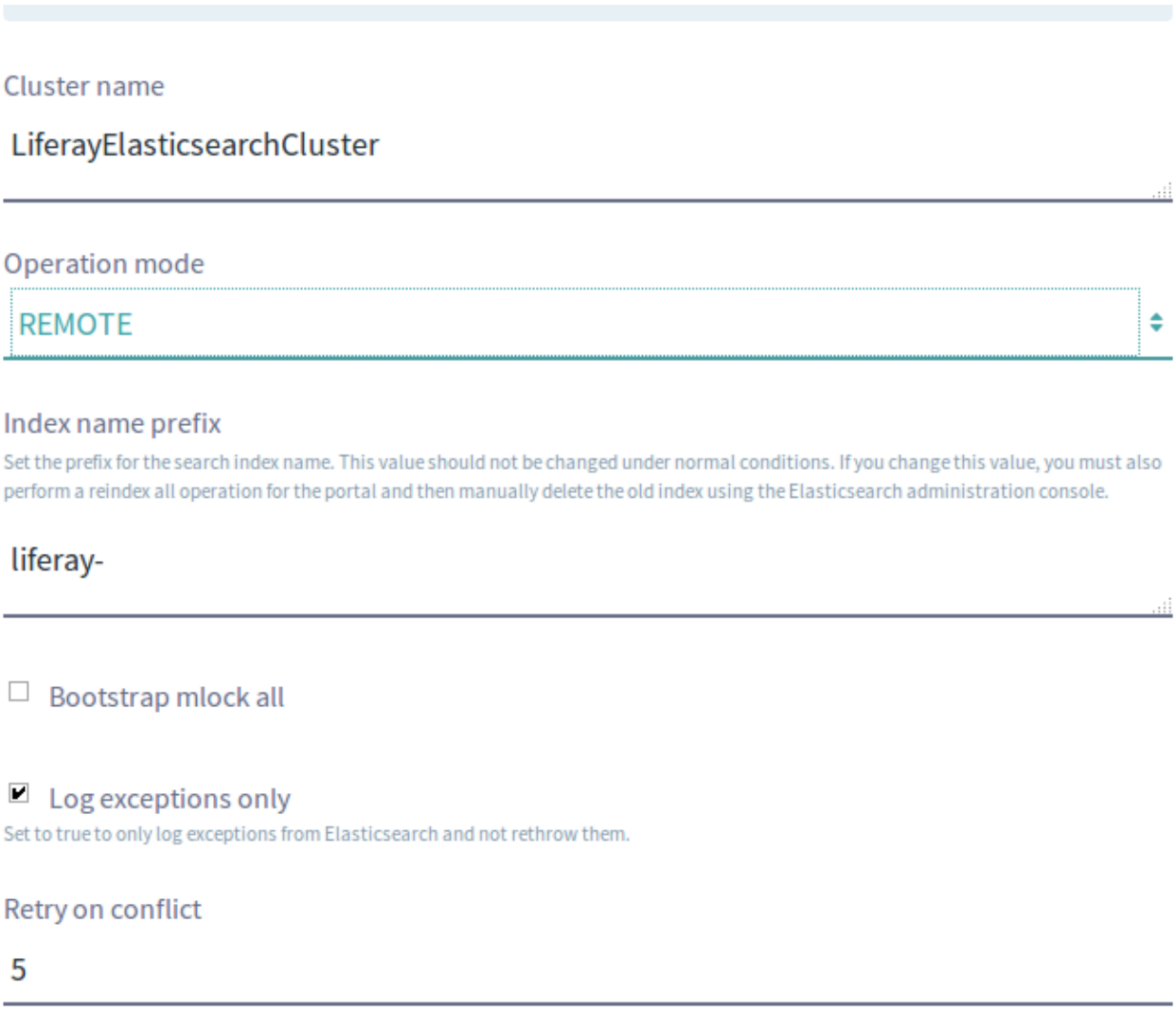


Figure 2: Set Operation Mode to *Remote* from System Settings.

5. After you switch operation modes (`EMBEDDED` → `REMOTE`), you must trigger a re-index. Navigate to *Control Panel* → *Server Administration*, find the *Index Actions* section, and click *Execute* next to *Reindex all search indexes.*

CONFIGURING THE ADAPTER WITH AN OSGI `.CONFIG` FILE

When preparing a system for production deployment, you want to set up a repeatable deployment process. Therefore, it's best to use the OSGi configuration file, where your configuration is maintained in a controlled source.

Follow these steps to configure the Elasticsearch adapter using an OSGi configuration file:

1. Create the following file to configure the default adapter (for Elasticsearch 2.4):

To configure the Liferay Connector to Elasticsearch 6, name your file thus:

```
[Liferay_Home]/osgi/configs/com.liferay.portal.search.elasticsearch6.configurati
```

2. Add this to the configuration file you just created:

```
operationMode="REMOTE"
# If running Elasticsearch from a different computer:
#transportAddresses="ip.of.elasticsearch.node:9300"
# Highly recommended for all non-prodcution usage (e.g., practice, tests, diagno
#logExceptionsOnly="false"
```

3. Start Liferay DXP or re-index if already running.

As you can see from the System Settings entry for Elasticsearch, there are a lot more configuration options available that help you tune your system for optimal performance. For a detailed accounting of these, refer to the reference article on Elasticsearch Settings.

What follows here are some known good configurations for clustering Elasticsearch. These, however, can't replace the manual process of tuning, testing under load, and tuning again, so we encourage you to examine the settings as well as the Elasticsearch documentation and go through that process once you have a working configuration.

Configuring a Remote Elasticsearch Host

In production systems Elasticsearch and Liferay DXP are installed on different servers. To make Liferay DXP aware of the Elasticsearch cluster, set

```
transportAddresses=[IP address of Elasticsearch Node]:9300
```

in the Elasticsearch adapter's OSGi configuration file. List as many or as few Elasticsearch nodes in this property as you'd like. This tells Liferay DXP the IP address or host name where search requests are to be sent. If using System Settings, set the value in the *Transport Addresses* property.

Note: In an Elasticsearch cluster you can list the transport addresses for multiple Elasticsearch nodes, if appropriate. Just use a comma-separated list in the `transportAddresses` property. If you set only one transport address, Liferay DXP loses contact with Elasticsearch if that node goes down.

host name or IP address Elasticsearch uses to communicate with other nodes). See here for more information.

Clustering Elasticsearch in Remote Operation Mode

Clustering Elasticsearch is easy. Each time you run the Elasticsearch start script, a new local storage node is added to the cluster. If you want four nodes running locally, for example, just run `./bin/elasticsearch` four times. If you only run the start script once, you have a cluster with just one node.

Elasticsearch 6.5: To start multiple local storage nodes in Elasticsearch 6.5, you must also configure `node.max_local_storage_nodes` to be something greater than `1`. See here for more information.

Elasticsearch's default configuration works for a cluster of up to ten nodes, since the default number of shards is `5`, while the default number of replica shards is `1`:

```
index.number_of_shards: 5
index.number_of_replicas: 1
```

Note: Elasticsearch uses the Zen Discovery Module by default, which provides unicast discovery. Additionally, nodes in the cluster communicate using the Transport Module, through TCP. See the Elasticsearch documentation for the available properties (to be set in the `elasticsearch.yml` file), and the Liferay DXP Elasticsearch Adapter's reference article for the adapter's available settings.

At a minimum, provide the list of hosts to act as gossip routers during unicast discovery in the `elasticsearch.yml`:

```
discovery.zen.ping.unicast.hosts: ["node1.ip.address", "node2.ip.address"]
```

Elasticsearch 6.5: Elasticsearch 6 removed the setting that specifies the number of shards and replicas in the `elasticsearch.yml` file. Configure these index-level settings in the Elasticsearch 6.5 connector, using the `indexNumberOfShards` and `indexNumberOfReplicas` properties to specify the number of primary shards and number of replica shards, respectively.

For more information on configuring an Elasticsearch cluster, see the documentation on Elasticsearch Index Settings.

Advanced Configuration of the Liferay Elasticsearch Adapter

While the Elasticsearch adapter has a lot of configuration options out of the box, you might find an Elasticsearch configuration you need that isn't provided by default. In this case, add the configuration options you need. If something is configurable for Elasticsearch, its configurable using the Elasticsearch adapter.

ADDING SETTINGS AND MAPPINGS TO THE LIFERAY ELASTICSEARCH ADAPTER

The available configuration options are divided into two groups: the ones you'll use most often by default, and a catch-all for everything else. So if the necessary setting isn't available by default, you can still configure it with the Liferay Elasticsearch adapter. Just specify the settings you need by using one or more of the `additionalConfigurations`, `additionalIndexConfigurations`, or `additionalTypeMappings` settings.

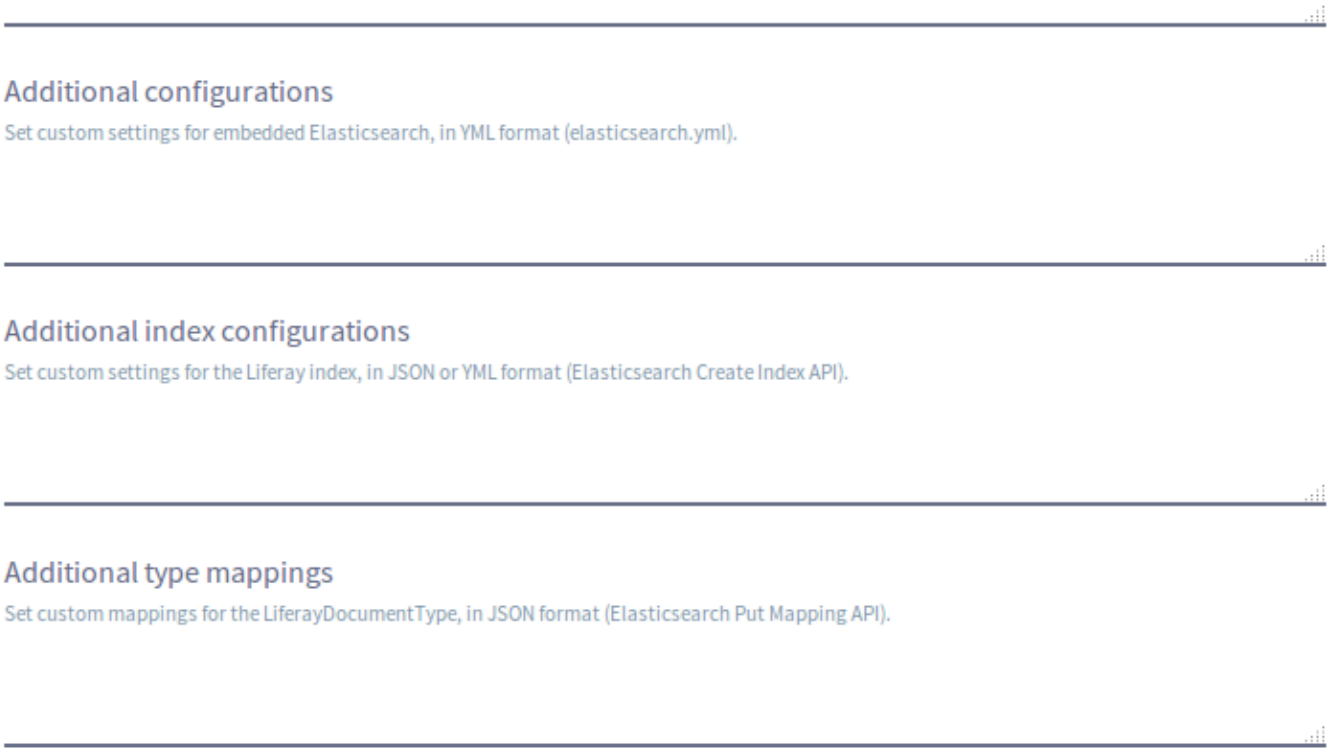


Figure 3: You can add Elasticsearch configurations to the ones currently available in System Settings.

Adding Configurations

`additionalConfigurations` is used to define extra settings (defined in YAML) for the embedded Elasticsearch or the local Elasticsearch client when running in remote mode. In production, only one additional configuration can be added here:

Elasticsearch 6.5: The Elasticsearch 6.5 connector includes the `client.transport.ping_timeout` as a native setting. Configure it through its dedicated setting rather than with `additionalConfigurations`.

The rest of the settings for the client are available as default configuration options in the Liferay Elasticsearch adapter. See the Elasticsearch Settings reference article for more information. See the Elasticsearch documentation for a description of all the client settings and for an example.

Adding Index Configurations

`additionalIndexConfigurations` is used to define extra settings (in JSON or YAML format) that are applied to the Liferay DXP index when it's created. For example, you can create custom analyzers and filters using this setting. For a complete list of available settings, see the Elasticsearch reference.

Here's an example that shows how to configure analysis that can be applied to a field or dynamic template (see below.

```
{
    "analysis": {
        "analyzer": {
            "kuromoji_liferay_custom": {
                "filter": [
                    "cjk_width",
                    "kuromoji_baseform",
                    "pos_filter"
                ],
                "tokenizer": "kuromoji_tokenizer"
            }
        },
        "filter": {
            "pos_filter": {
                "type": "kuromoji_part_of_speech"
            }
        }
    }
}
```

Adding Type Mappings

and here. Use `additionalTypeMappings` for new field (`properties`) and dynamic template mappings, but do not try to override existing mappings. If any of the mappings set here overlap with existing mappings, index creation will fail. Use `overrideTypeMappings` to replace the default mappings.

As with dynamic templates, you can add sub-field mappings to Liferay DXP's type mapping. These are referred to as properties in Elasticsearch.

```
{
    "LiferayDocumentType": {
        "properties": {
            "fooName": {
                "index": "not_analyzed",
                "store": "yes",
                "type": "string"
            }
        }
    }
}
```

Elasticsearch 6: The above property mapping looks different in Elasticsearch 6.1:

```
{
    "LiferayDocumentType": {
        "properties": {
            "fooName": {
                "index": "true",
                "store": "true",
                "type": "keyword"
            }
        }
    }
}
```

See here for more details on Elasticsearch's field datatypes.

The above example shows how a `fooName` field might be added to Liferay DXP's type mapping. Because `fooName` is not an existing property in the mapping, it will work just fine. If you try to override an existing

To see that your additional mappings have been added to the `LiferayDocumentType`, navigate to this URL after saving your additions and reindexing:

```
http://[HOST]:[ES_PORT]/liferay-[COMPANY_ID]/_mapping/LiferayDocumentType?pretty
```

Here's what it would look like for an Elasticsearch instance running on `localhost:9200`, with a Liferay DXP Company ID of `20116`:

```
http://localhost:9200/liferay-20116/_mapping/LiferayDocumentType?pretty
```

In the above URL, `liferay-20116` is the index name. Including it indicates that you want to see the mappings that were used to create the index with that name.

Overriding Type Mappings

Use `overrideTypeMappings` to override Liferay DXP's default type mappings. This is an advanced feature that should be used only if strictly necessary. If you set this value, the default mappings used to define the Liferay Document Type in Liferay DXP source code (for example, `liferay-type-mappings.json`) are ignored entirely, so include the whole mappings definition in this property, not just the segment you're modifying. To make a modification, find the entire list of the current mappings being used to create the index by navigating to the URL

```
http://[HOST]:[ES_PORT]/liferay-[COMPANY_ID]/_mapping/LiferayDocumentType?pretty
```

Copy the contents in as the value of this property (either into System Settings or your OSGi configuration file). Leave the opening curly brace `{`, but delete lines 2-4 entirely:

```
"liferay-[COMPANY_ID]": {
    "mappings" : {
        "LiferayDocumentType" : {
```

Then, from the end of the mappings, delete the concluding three curly braces.

```
        }
    }
```

Now modify whatever mappings you'd like. The changes take effect once you save the changes and trigger a reindex from Server Administration. If you need to add new custom mappings without overriding any defaults, use `additionalTypeMappings` instead.

Here's a partial example, of a dynamic template that uses the analysis configuration above to analyze all string fields that end with `_ja` , overriding the default `template_ja` mapping.

```
{
    "LiferayDocumentType": {
        "dynamic_templates": [
            {
                "template_ja": {
                    "mapping": {
                        "analyzer": "kuromoji_liferay_custom",
                        "index": "analyzed",
                        "store": "true",
                        "term_vector": "with_positions_offsets",
                        "type": "string"
                    },
                    "match": "\\w+_ja\\b|\\w+_ja_[A-Z]{2}\\b",
                    "match_mapping_type": "string",
                    "match_pattern": "regex"
                }
                ...
            }
        ]
    }
}
```

Note: There's actually a third way to add configuration options to the Elasticsearch adapter. You or your favorite developer can publish a Settings Contributor Component and deploy it to Liferay's OSGi runtime. A developer tutorial on Search Extension Points will be written for this. In summary, the contributor module needs these things:

- A class that implements either `com.liferay.portal.search.elasticsearch.settings.SettingsContributor` or `com.liferay.portal.search.elasticsearch.settings.IndexSettingsContributor` .
    - If you're adding settings that would go into `additionalConfigurations` , override `SettingsContributor` .

- An OSGi `@Component` annotation for either of the implementations mentioned in the last step.

MULTI-LINE YAML CONFIGURATIONS

If you configure the settings from the last section using an OSGi configuration file, you might find yourself needing to write YAML snippets that span multiple lines. The syntax for that is straightforward and just requires appending each line with `\n\` , like this:

```
additionalConfigurations=\
                    cluster.routing.allocation.disk.threshold_enabled: false\n\
                    cluster.service.slow_task_logging_threshold: 600s\n\
                    index.indexing.slowlog.threshold.index.warn: 600s\n\
                    index.search.slowlog.threshold.fetch.warn: 600s\n\
                    index.search.slowlog.threshold.query.warn: 600s\n\
                    monitor.jvm.gc.old.warn: 600s\n\
                    monitor.jvm.gc.young.warn: 600s
```

Troubleshooting Elasticsearch

Sometimes things don't go as planned. If you've set up Liferay DXP with Elasticsearch in remote mode, but Liferay DXP can't connect to Elasticsearch, check these things:

- Cluster name: The value of the `cluster.name` property in Elasticsearch must match the `clusterName` property you configured for Liferay's Elasticsearch adapter.

- Transport address: The value of the `transportAddress` property in the Elasticsearch adapter must match the port where Elasticsearch is running. If Liferay DXP is running in embedded mode, and you start a standalone Elasticsearch node or cluster, it detects that port `9300` is taken and switches to port `9301` . If you then set Liferay's Elasticsearch adapter to remote mode, it continues to look for Elasticsearch at the default port (`9300`).

Now you have Elasticsearch configured for use. If you're a Liferay DXP customer, you can read here to learn about configuring Shield to secure your Elasticsearch data.

Elasticsearch Connector System Settings, By Operation Mode

Adapter Setting/Operation Mode	EMBEDDED	REMOTE
clusterName	x	x
operationMode	x	x
indexNamePrefix	x	x
indexNumberOfReplicas*	x	x
indexNumberOfShards*	x	x
bootstrapMlockAll	x	-
logExceptionsOnly	x	x
retryOnConflict	x	x
discoveryZenPingUnicastHostsPort	x	-
networkHost	x	-
networkBindHost	x	-
networkPublishHost	x	-
transportTcpPort	x	-
transportAddresses	-	x
clientTransportSniff	-	x
clientTransportIgnoreClusterName	-	x
clientTransportPingTimeout*	-	x

clientTransportNodesSamplerInterval	-	x
httpEnabled	x	-
httpCORSEnabled	x	-
httpCORSAllowOrigin	x	-
httpCORSConfigurations	x	-
additionalConfigurations	x	x
additionalIndexConfigurations	x	x
additionalTypeMappings	x	x
overrideTypeMappings	x	x

* **Note:** Available in the Liferay Connector to Elasticsearch 6 only.

Related Topics

Introduction to Liferay Search

Customizing Liferay Search

1 This is, of course, a nod to all those fans of Boaty Mcboatface.

« Preparing to Install Elasticsearch Securing Elasticsearch with Shield »

¿Fue útil este artículo?

Sí No

Usuarios a los que les pareció útil: 1 de 1

¿Necesitas más ayuda?

Abre un ticket

Contacta con nosotros

Pregunta en la Comunidad

Liferay University
Forma a tu equipo en los puntos esenciales del uso de Liferay con videos detallados y ejercicios prácticos diseñados para simular una formación presencial

Liferay Global Services
Aumenta tu conocimiento sobre el producto y el desarrollo gracias a expertos que te ayudarán a maximizar tu ROI