

IndexWriters

- [Index writers in Nutch](#)
- [Structure of index-writers.xml](#)
 - [Mapping section](#)
 - [Use case](#)
 - [Parameters section](#)
 - [Solr indexer properties](#)
 - [Rabbit indexer properties](#)
 - [Dummy indexer properties](#)
 - [Elasticsearch indexer properties](#)
 - [Elasticsearch rest indexer properties](#)
 - [CloudSearch indexer properties](#)
 - [CSV indexer properties](#)

Index writers in Nutch

An index writer is a component of the indexing job, which is used for sending documents from one or more segments to an external server. In Nutch, these components are found as plugins. Nutch includes these out-of-the-box indexers:

Indexer	Description
indexer-solr	Indexer for a Solr server
indexer-rabbit	Indexer for a RabbitMQ server
indexer-dummy	Indexer usually used for debugging, it writes in a plain text file
indexer-elastic	Indexer for an Elasticsearch server
indexer-elastic-rest	Indexer for Elasticsearch, but using Jest to connect with the REST API provided by Elasticsearch
indexer-cloudsearch	Indexer for Amazon CloudSearch
indexer-csv	Indexer for writing documents to a CSV file

Structure of index-writers.xml

The configuration for the indexers is in the index-writers.xml file, included in the official Nutch distribution. The structure of this file is quite simple and consists mainly of a list of indexers (<writer> element):

```
<writers>
<writer id="[writer_id]" class="[implementation_class]">
<mapping>
...
</mapping>
<parameters>
...
</parameters>
</writer>
...
</writers>
```

Each <writer> element has two mandatory attributes:

1. [writer_id] is a unique identification for each configuration. This feature allows Nutch to distinguish each configuration, even when they are for the same index writer. In addition, it allows to have multiple instances for the same index writer, but with different configurations.
2. [implementation_class] corresponds to the canonical name of the class that implements the [IndexWriter](#) extension point. For the indexers provided by Nutch out-of-the-box the possible values of [implementation_class] are:

Indexer	Implementation class
indexer-solr	org.apache.nutch.indexwriter.solr.SolrIndexWriter
indexer-rabbit	org.apache.nutch.indexwriter.rabbit.RabbitIndexWriter
indexer-dummy	org.apache.nutch.indexwriter.dummy.DummyIndexWriter

indexer-elastic	org.apache.nutch.indexwriter.elastic.ElasticIndexWriter
indexer-elastic-rest	org.apache.nutch.indexwriter.elasticrest.ElasticRestIndexWriter
indexer-cloudsearch	org.apache.nutch.indexwriter.cloudsearch.CloudSearchIndexWriter
indexer-csv	org.apache.nutch.indexwriter.csv.CSVIndexWriter

Each `<writer>` element contains two child elements: `<mapping>` and `<parameters>`

Mapping section

The `<mapping>` element is independent for each configuration and it's where you configure the modifications that will be applied to each document before it is sent to its final destination. The `<mapping>` element contains 3 child elements: `<copy>`, `<rename>` and `<remove>`

- `<copy>` indicates which fields should be copied from the document and to which field they should be copied. Each child element of `<copy>` element, has this form: `<field source="[source]" dest="[destination]" />`
 - `[source]` indicates the field's name to be copied.
 - `[destination]` indicates to which field or fields should be copied. The value of this attribute can be a comma separated list. In this case, the value of **source** attribute will be copied into each field in the list. For example: if the configuration is `<field source="title" dest="description,search" />`, the value of the **title** field will be copied for the **description** and **search** fields.
- `<rename>` indicates which fields of the document should be renamed. Each child element of `<rename>` element, has this form: `<field source="[source]" dest="[destination]" />`
 - `[source]` indicates the field's name to be renamed.
 - `[destination]` indicates the new name of the field. For example: if the configuration is `<field source="metatag.description" dest="description" />`, the field **metatag.description** will be renamed as **description**.
- `<remove>` indicates which fields of the document should be removed. Each child element of `<remove>` element, has the form: `<field source="[source]" />`
 - `[source]` indicates the field's name to be remove.



If you don't want to modify the document, just leave `<copy>`, `<rename>` and `<remove>` empty, like: `<mapping> <copy /> <rename /> <remove /> </mapping>`

Use case

We have two servers previously configured (Solr and RabbitMQ). We want to send documents to each one, but with a different structure. Prior to the index step, each document has this hypothetical structure:

```
host: "www.example.org"
domain: "example.org"
title: "Example page"
metatag.description: "Example page description"
metatag.keywords: ["example", "page"]
segment: 20180621163128
```

With this configuration we modify the structure of each document in different ways, depending the index writer:

```

<writer id="indexer_solr_1" class="org.apache.nutch.indexwriter.solr.SolrIndexWriter">
<parameters>
<!-- Parameters here -->
</parameters>
<mapping>
<copy/>
<rename>
<field source="metatag.description" dest="description"/>
<field source="metatag.keywords" dest="keywords"/>
</rename>
<remove>
<field source="segment"/>
</remove>
</mapping>
</writer>
<writer id="indexer_rabbit_1" class="org.apache.nutch.indexwriter.rabbit.RabbitIndexWriter">
<parameters>
<!-- Parameters here -->
</parameters>
<mapping>
<copy>
<field source="title" dest="search"/>
</copy>
<rename>
<field source="metatag.description" dest="description"/>
<field source="metatag.keywords" dest="keywords"/>
<field source="domain" dest="domain_name"/>
</rename>
<remove />
</mapping>
</writer>

```

For **indexer-solr** we'll get documents like:

```

host: "www.example.org"
domain: "example.org"
title: "Example page"
description: "Example page description"
keywords: ["example", "page"]

```

For **indexer-rabbit** the document's structure is like:

```

host: "www.example.org"
domain_name: "example.org"
title: "Example page"
search: "Example page"
description: "Example page description"
keywords: ["example", "page"]
segment: 20180621163128


```

Parameters section


The `<parameters>` element is independent for each configuration and is where the parameters that the indexer needs are specified. Each parameter has the form `<param name="[name]" value="[value]"/>` and the values it can take depend on the indexer that you want to configure. Below is a description of the arguments of each indexer provided by Nutch out-of-the-box individually.

Solr indexer properties

Parameter Name	Description	Default value
type	Specifies the SolrClient implementation to use. This is a string value of one of the following cloud or http . The values represent CloudSolrServer or HttpSolrServer respectively.	http

url	Defines the fully qualified URL of Solr into which data should be indexed. Multiple URL can be provided using comma as a delimiter. When the value of <code>type</code> property is cloud , the URL should not include any collections or cores; just the root Solr path.	http://localhost:8983/solr/nutch
collection	The collection used in requests. Only used when the value of <code>type</code> property is cloud .	
weight.field	Field's name where the weight of the documents will be written. If it is empty no field will be used.	
commit Size	Defines the number of documents to send to Solr in a single update batch. Decrease when handling very large documents to prevent Nutch from running out of memory. <div>  It does not explicitly trigger a server side commit. </div>	1000
auth	Whether to enable HTTP basic authentication for communicating with Solr. Use the <code>username</code> and <code>password</code> properties to configure your credentials.	false
username	The username of Solr server.	username
password	The password of Solr server.	password

Rabbit indexer properties

Parameter Name	Description	Default value
server.uri	URI with connection parameters in the form <code>amqp://<username>:<password>@<hostname>:<port>/<virtualHost></code> Where: <ul style="list-style-type: none"> <code><username></code> is the username for RabbitMQ server. <code><password></code> is the password for RabbitMQ server. <code><hostname></code> is where the RabbitMQ server is running. <code><port></code> is where the RabbitMQ server is listening. <code><virtualHost></code> is where the exchange is and the user has access. 	<code>amqp://guest:guest@localhost:5672/</code>
binding	Whether the relationship between an exchange and a queue is created automatically. <div>  Binding between exchanges is not supported. </div>	false
binding.arguments	Arguments used in binding. It must have the form <code>key1=value1,key2=value2</code> . This value is only used when the exchange's type is headers and the value of <code>binding</code> property is true . In other cases is ignored.	
exchange.name	Name for the exchange where the messages will be sent.	
exchange.options	Options used when the exchange is created. Only used when the value of <code>binding</code> property is true . It must have the form <code>type=<type>,durable=<durable></code> Where: <ul style="list-style-type: none"> <code><type></code> is direct, topic, headers or fanout <code><durable></code> is true or false 	<code>type=direct,durable=true</code>
queue.name	Name of the queue used to create the binding. Only used when the value of <code>binding</code> property is true .	<code>nutch.queue</code>

queue. options	Options used when the queue is created. Only used when the value of <code>binding</code> property is true . It must have the form <code>durable=<durable>,exclusive=<exclusive>,auto-delete=<auto-delete>,arguments=<arguments></code> Where: <ul style="list-style-type: none"> <code><durable></code> is true or false <code><exclusive></code> is true or false <code><auto-delete></code> is true or false <code><arguments></code> must be the form <code>key1:value1;key2:value2</code> 	durable=true, exclusive=false,auto-delete=false
routingkey	The routing key used to route messages in the exchange. It only makes sense when the exchange type is topic or direct .	Value of <code>queue.name</code> property
commit. mode	single if a message contains only one document. In this case, a header with the action (write, update or delete) will be added. multiple if a message contains all documents.	multiple
commit. size	Amount of documents to send into each message if the value of <code>commit.mode</code> property is multiple . In single mode this value represents the amount of messages to be sent.	250
headers. static	Headers to add to each message. It must have the form <code>key1=value1,key2=value2</code> .	
headers. dynamic	Document's fields to add as headers to each message. It must have the form <code>field1,field2</code> . Only used when the value of <code>commit.mode</code> property is single	

Dummy indexer properties

Parameter Name	Description	Default value
path	Path where the file will be created.	./dummy-index.txt
delete	If delete operations should be written to the file.	false

Elasticsearch indexer properties

Parameter Name	Description	Default value
host	Comma-separated list of hostnames to send documents to using TransportClient . Either host and port must be defined or cluster.	
port	The port to connect to using TransportClient .	9300
cluster	The cluster name to discover. Either host and port must be defined or cluster.	
index	Default index to send documents to.	nutch
max.bulk.docs	Maximum size of the bulk in number of documents.	250
max.bulk.size	Maximum size of the bulk in bytes.	2500500
exponential.backoff.millis	Initial delay for the BulkProcessor exponential backoff policy.	100
exponential.backoff.retries	Number of times the BulkProcessor exponential backoff policy should retry bulk operations.	10
bulk.close.timeout	Number of seconds allowed for the BulkProcessor to complete its last operation.	600

Elasticsearch rest indexer properties

Parameter Name	Description	Default value
host	The hostname or a list of comma separated hostnames to send documents to using Elasticsearch Jest. Both host and port must be defined.	
port	The port to connect to using Elasticsearch Jest.	9200

index	Default index to send documents to.	nutch
max.bulk.docs	Maximum size of the bulk in number of documents.	250
max.bulk.size	Maximum size of the bulk in bytes.	2500500
user	Username for auth credentials (only used when https is enabled)	user
password	Password for auth credentials (only used when https is enabled)	password
type	Default type to send documents to.	doc
https	true to enable https, false to disable https. If you've disabled http access (by forcing https), be sure to set this to true, otherwise you might get "connection reset by peer".	false
trustlocalhostnames	true to trust elasticsearch server's certificate even if its listed domain name does not match the domain they are hosted or false to check if the elasticsearch server's certificate's listed domain is the same domain that it is hosted on, and if it doesn't, then fail to index (only used when https is enabled)	false
languages	A list of strings denoting the supported languages (e.g. <code>en</code> , <code>de</code> , <code>fr</code> , <code>it</code>). If this value is empty all documents will be sent to <code>index</code> property. If not empty the Rest client will distribute documents in different indices based on their <code>languages</code> property. Indices are named with the following schema: <code>index separator language</code> (e.g. <code>nutch_de</code>). Entries with an unsupported <code>languages</code> value will be added to <code>index index separator sink</code> (e.g. <code>nutch_others</code>).	
separator	Is used only if <code>languages</code> property is defined to build the index name (i.e. <code>index separator lang</code>).	–
sink	Is used only if <code>languages</code> property is defined to build the index name where to store documents with unsupported languages (i.e. <code>index separator sink</code>).	others

CloudSearch indexer properties

Parameter Name	Description	Default value
endpoint	Endpoint where service requests should be submitted.	
region	Region name.	
batch.dump	true to send documents to a local file.	false
batch.maxSize	Maximum number of documents to send as a batch to <code><<GetText(CloudSearch)>></code> .	-1

CSV indexer properties

Parameter Name	Description	Default value
fields	Ordered list of fields (columns) in the CSV file	id,title,content
charset	Encoding of CSV file	UTF-8
separator	Separator between fields (columns)	,
valuesep	Separator between multiple values of one field	
quotechar	Quote character used to quote fields containing separators or quotes	"
escapechar	Escape character used to escape a quote character	"

maxfieldlength	Max. length of a single field value in characters	4096
maxfieldvalues	Max. number of values of one field, useful for, e.g., the anchor texts field	12
header	Write CSV column headers	true
outpath	Output path / directory (local filesystem path, relative to current working directory)	csvindexwriter



The CSV indexer does not work in distributed mode, the output is written to the local filesystem, not to HDFS, see [NUTCH-1541](#).