Lucid.anda Connector Framework

Reference

Table of Contents

Lucid.anda is a general framework for efficient traversal of data repositories with a rich set of configuration properties that allow fine-grained control of the kind, amount, and rate of data retrieval. Specific implementations have different configuration properties according to the repository type. To see which properties are required/optional, query the REST API via the URL: api/connectors/plugins/lucid.anda/types/<connector-type>, e.g. to see the properties available for lucid.anda-web plugin using the `curl` command-line HTTP client, sent a GET request along with Fusion user and password:

```
http://<server>:<port>/api/connectors/plugins/lucid.anda/types/web
```

Copy

Basics Configuration Properties

The set of "Basics" configuration properties limit the scope of the crawl.

The crawler fetches the contents of the specified startLink property. Any links found in the contents are added to the set of links to traverse. The connector keeps track of nodes it has seen in a database known as "crawldb" to prevent re-processing. This database tracks nodes which have been indexed, as well as which nodes have been found to be redirects, duplicates, or otherwise aliases of another node.

API Name / UI Label	Description
startLinks / Start Links *required*	A list of URIs to use as the seed URIs for the crawl. Changing this field after crawling the content will require you to clear the crawldb.
diagnosticMode / Diagnostic mode? *optional*	If **true**, diagnostic information is written to the `connectors.log`: ● excluded links and the reason for exclusion ● if `dedupeField` or `dedupeScript` is enabled, the signature strings for each item are printed ● if a `rewriteLinkScript` is configured, the re-written links are printed The default is **false**.
restrictToTree / Restrict to tree? *optional*	If **true**, the default, the crawler will restrict the crawl to only the tree of items below the provided startLinks.
depth / Max depth Changing this field after crawling the content will require you to clear the crawldb. *optional*	The number of path levels to descend. The default, **-1**, indicates unlimited depth, and will crawl all URIs that match other definitions of the crawl. Changing this field after crawling the content will require you to clear the crawldb.
maxItems / Max items *optional*	Defines the maximum number of items to retrieve during a crawl. This can be used to limit the crawl of a very large dataset to a smaller number of documents in order to gauge performance or to test pipeline settings. If this setting is modified mid-crawl (i.e., a crawl is started, then stopped before it finishes, then starts again), the original value will be retained. If a crawl is allowed to finish and then this property is decreased, subsequent re-crawls will respect the new value, but the specific documents items retrieved will be an unpredictable subset of the original document set. The default is **-1**, to retrieve all documents found that are allowed according to other property definitions.
includeExtensions / Included file-extensions *optional*	Defines a list of file extensions to include in the crawl. Changing this field after crawling the content will require you to clear the crawldb.

excludeExtensions / Excluded file-extensions *optional*	Defines a list of file extensions to exclude from the crawl. Only the extension is necessary with no additional characters, as in `pdf` or `.pdf` . Changing this field after crawling the content will require you to clear the crawldb.
excludeRegexes / Exclusive regexes *optional*	Defines a list of regular expressions to exclude specific URIs or URI patterns from the crawl. Changing this field after crawling the content will require you to clear the crawldb.
chunkSize / Chunk size *optional*	The number of items to batch for each round of fetching. The default is **50** items.
fetchThreads / Fetch threads *optional*	The number of fetch threads. The default is **5** threads.
fetchDelayMS / Fetch delay (ms) *optional*	The number of milliseconds to wait between document requests. This property can be used to throttle a crawl in cases where too frequent requests may cause performance issues in the crawled website and the site does not have a robots.txt file in place to control incoming requests from automated agents. The default is **0** milliseconds.
emitThreads / Emit threads *optional*	The number of emit threads. The emitter is responsible for the output of documents from the crawler to Fusion. The default is **5** threads.
delete / Enable Deletion? *optional*	If **true**, the default, documents will be removed from the index if they are considered "defunct". There are two cases when a document will be considered defunct: ● A document (A) used to have content and now redirects to another document that already exists in the index (B). In this case, document A will be removed in favor of document B. ● A document fails to be fetched because of a 404, a 500 error, network timeout, or several other possible causes of failure. In this case, the deleteErrorsAfter property is also used to indicate the number of failures to allow before removing the document from the index.
deleteErrorsAfter / Delete fetch-failures after...? *optional*	The number of fetch failures before a document is removed from the index. The default is **-1**, which means documents that return errors on recrawl will never be removed. If you would like document removed after a specific threshold, set this property to your desired threshold.

Fetcher Configuration Properties

Fetcher configuration properties vary by plugin. Fetcher configuration properties are distinguished by prefix "f.", e.g. "f.maxSizeBytes".

timeout (ms) *optional*	The default is **10000** milliseconds, or 10 seconds.
f.maxSizeBytes / Max file size (bytes) *optional*	Defines the maximum size of a document to crawl, expressed in bytes. Documents larger than this will be dropped from the crawl. The default is 5Mb (**4,194,304 bytes**) per document.
f.proxy / HTTP proxy (<host>:<port> format) *optional*	The location of the HTTP proxy, if any. The proxy address should be expressed in `host:port` format.
f.allowAllCertificates / Allow all HTTPS certificates? *optional*	Boolean value, default is false. If true, this disables security checks against SSL/TLS certificate signers and origins by skipping the hostname-verification logic. This allows certificates signed by anyone, including self-signed certificates. Hostname-verification logic restricts access to only those certificates which are signed by certificate authorities and certificates in the keystore.
f.credentialsFile / Authentication credentials filename *optional*	The name of the file within the crawler-container directory that contains authentication credentials. This file is in JSON format and should be located in `{fusion_path}/connectors/container/lucid.anda/datasourceID` , where 'datasourceID' is the ID you have given to the datasource that will use the file. See also the section Website Connector and Datasource Configuration lucid.anda-web for more details about this file and the properties it should contain.
f.sitemapURLs / Sitemap URLs *optional*	A list of URLs that are sitemaps. The URLs added with this property, and all URLs found in each sitemap, will be added to the list of start links for the datasource and crawled accordingly. A sitemap URL that is a sitemap index, or a sitemap that links other sitemaps, is also supported. Each URL found in each linked sitemap will be crawled in accordance with other include or exclude rules of the crawl. If the datasource should only contain a sitemap as the main start link, the sitemap URL should be provided to both the start link property and also to the sitemap property. Sitemaps will only be treated as sitemaps when the URL is provided as part of this property. When using the REST API, the sitemaps should be provided as a list, such as: `"f.sitemapURLs" : [ "http://site.com/sitemap1.html", "http://site.com/sitemap2.xml" ]`
f.obeyRobots / Obey robots.txt? *optional*	Boolean value, default is true. If **true,** the Allow, Disallow and other directives found in robots.txt will be respected.
f.obeyRobotsDelay / Obey robots.txt Crawl-delay? *optional*	Boolean value, default is true. If **true**, crawl-delay directives found in robots.txt will be respected.
f.appendTrailingSlashToLinks / Append a trailing slash to link URLs? *optional*	Boolean value, default is false. If **true**, a trailing slash ('/') will be added to URLs when the link does not end in a dot ('.').

f.defaultCharSet / Default character set *optional*	Name of default character set. Default is UTF-8
f.defaultMIMEType / Default MIME type *optional*	Name of default MIME type. Default is application/octet-stream.
f.respectMetaEquivRedirects / Respect <meta http-equiv=\"refresh\" /> redirects? *optional*	Boolean value, default is false. If true, the web-crawler will respect <meta http-equiv=\"refresh\" /> redirects embedded in the <head /> tag of source HTML itself, e.g.: `<meta http-equiv="refresh" content="0; url=http://example.com/">` Copy - -
f.userAgentName / HTTP user-agent name *optional*	The name to provide as the User-Agent name in HTTP request. The default is **Lucidworks-Anda/1.0**.
f.userAgentEmail / HTTP user-agent email address *optional*	An email address to pass with the user-agent information while crawling. The default is empty.
f.userAgentWebAddr / HTTP user-agent web address *optional*	A web address to use as a HTTP user-agent web address. The default is empty.

Content Filtering and Selection Configuration Properties

These properties are only used by the web plugin. Like the fetcher properties names, they have the prefix "f".

API Name / UI Label	Description
f.filteringRootTags / Root elements to filter *optional*	A list of HTML root elements whose child-elements will be used to extract the website content. The default list includes **body** and **head**.
f.scrapeLinksBeforeFiltering / Scrape links before filtering? *optional*	If **true**, content will be checked for links before it is filtered of other elements in accordance with other include/exclude rules. The default is **false**, which means links will be extracted after other elements have been filtered.

	small list of known tags you know you want to include but also want to exclude all other tags.
f.includeTagClasses / HTML tag-classes to include *optional*	A list of HTML tag classes of elements to include in the crawled content.
f.includeTagIDs / HTML tag-IDs to include *optional*	A list of the HTML tag IDs of elements to include in the crawled content.
f.includeSelectors / Jsoup inclusive selectors *optional*	A list of Jsoup selectors for elements to include in the crawled content. Jsoup allows using a CSS-like query syntax to find matching elements. For more information on Jsoup selectors, see the Jsoup Cookbook section on Jsoup selector syntax .
f.excludeTags / HTML tags to exclude *optional*	A list of HTML tag names for elements to exclude from the crawled documents.
f.excludeTagClasses / HTML tag-classes to exclude *optional*	A list of HTML tag classes of elements to exclude from the crawled content.
f.excludeTagIDs / HTML tag-IDs to exclude *optional*	A list of the HTML tag IDs of elements to exclude from the crawl.
f.excludeSelectors / Jsoup exclusive selectors *optional*	A list of jsoup selectors for elements to exclude from the crawled content. For more information on Jsoup selectors, see the Jsoup Cookbook section on Jsoup selector syntax .
f.tagFields / HTML tag fields *optional*	A list of HTML tag names for elements that will be added to their own fields. The new field will have the same name as the tag defined.
f.tagIDFields / HTML tag-ID fields *optional*	A list of HTML tag IDs for elements that will be added to their own fields. The new field will have the same name as the tag ID defined.
f.tagClassFields / HTML tag-class fields *optional*	A list of HTML tag classes for elements that will be added to their own fields. The new field will have the same name as the tag class defined.
f.selectorFields / Jsoup selector fields *optional*	A list of selectors in Jsoup format to put content into its own field. This property allows you to extract HTML tag elements and put them in their own field. Such as, 'h1' would make a field on each document with the content of the h1 tag on each page. You can then use field mapping in the index pipeline to copy that content to another field as appropriate for your schema. For more information on Jsoup selectors, see the Jsoup Cookbook section on Jsoup selector syntax . In Fusion v1.1, this property was renamed from f.fieldSelectors to f.selectorFields.

crawls after the first complete crawl, and the default refresh policy is to simply re-crawl all items. The refreshAll property is true by default to create that behavior, so the first step in configuring a refresh-policy is to set refreshAll to false.

There are five types of refresh policies: "refreshStartLinks", "refreshErrors", "refreshOlderThan", "refreshIdPrefixes", "refreshIDRegexes".

This is scriptable via a JavaScript function supplied as property "refreshScript", e.g.:

```
function shouldRefresh(id, depth, lastModified, lastFetched, lastEmitted, error) {
  if (null !== error) {
    if (null !== error.getCause()) {
      if (-1 !== error.getCause().getMessage().indexOf("503")) {
        return true;
      }
    }
  }
  return false;
}
```

Copy

API Name / UI Label	Description
refreshAll *optional*	Boolean value, default is true. If true, re-crawl all items.
refreshStartLinks *optional*	Refresh all items specified in property "startLinks".
refreshErrors *optional*	Refresh all items that failed in any way last time
refreshOlderThan *optional*	Refresh all items whose last-fetched-date is older than this property’s value, **in seconds**. e.g. use 86400 to refresh all items that have not been fetched in one day or more
refreshIdPrefixes *optional*	An array of strings of prefixes. Refresh all items whose ID begins with any of these prefixes, e.g. "http://lucidworks.com/product/" to only refresh product pages in a crawl of a web-site.
refreshIDRegexes	An array of strings of regexes. Refresh all items which match any regex, e.g., ".**/product/.**\.html" to

forceRefresh / Force refreshing? *optional*	Boolean value, default is false. If true, re-crawl all items, even if they have not changed since last crawl. If you make a change to your pipeline or schema that will lead to analyzing/indexing the text differently, you would want to recrawl all items. forceRefresh is different from clearing the datasource because it allows you to clear the last-modified date and ETag while retaining its history.

Dedupe Configuration Properties

Fusion can be configured to deduplicate documents based on:

- the entire contents of the document
- the contents of a specified field
- custom deduplication based on a document signature generated by a user-supplied JavaScript function genSignature() which returns a string. The Fusion UI Admin tool provides a JavaScript-aware input box which so that you can create and edit this function directly in Fusion.

Dedupe works by maintaining a signature for each document, and ensuring that exactly **one** document appears in Solr for **each signature**. It does this by designating the first document it encounters with a particular signature, making it the "canonical" document. All subsequent documents with that signature are designated as "aliases".

It keeps track of the current canonical document for a particular signature across crawls, and when a document signature changes, it maintains its guarantee that exactly one document with each signature shows up in Solr.

In the case where custom deduplication is done either using a field or a custom signature, you must specify either the field or the JavaScript function, accordingly. The value of this string is found in the `dedupeSignature_s` field.

If the property "dedupe" (UI control checkbox "Dedupe on Content") is true but neither a field or JavaScript function are specified, the raw contents of the document are used for deduplication. No deduplication signature is generated, therefore the resulting document does not have a `dedupeSignature_s` field.

Here is an example of a `genSignature()` function:

```
function genSignature(content) {
    var signature = "";
    if (content.hasField("h2")) {
        }
    }
    return signature.length > 0 ? signature : null;
}
```

Copy

This example finds duplicates based on the h2 fields in each document. This script assumes that the h2 headers in the documents have been pulled into a field with the 'f.fieldSelectors' property. The entire content object is available here, so implementations of this class can dedupe on any combination of fields. The `genSignature()` function should return null when the fields needed to generate a signature are not present.

API Name / UI Label	Description
dedupe / Dedupe on content? *optional*	Boolean value, default is false. If **true**, the crawler will try to de-duplicate content. This can be done with an analysis of the raw content of the document, or based on content in a specific named field (dedupeField) or with JavaScript (dedupeScript). If a document is identified as a duplicate of another, the URI for the duplicate document will be entered into the crawl database as an alias.
dedupeSignatureString / Save the dedupe signature string? *optional*	Boolean value, default is false. If true, the deduplication signature string will be saved as part of the Solr document in the field 'dedupeSignature_s', so that users can see the string used for deduplication. This string can be very long, and may cause Solr to throw an error about an "immense" term.
dedupeField / Dedupe field *optional*	A field to use in de-duplication. If no field is defined, and no JavaScript is defined with dedupeScript, the item's full raw-content will be used by default.
dedupeScript / Dedupe script *optional*	Specifies a JavaScript to perform custom de-duplication. The JavaScript should contain a `genSignature()` function to ensure proper functioning.

Splitter Configuration Properties

These properties determine how to process .csv and .tsv files.

API Name / UI Label	Description
splitCSV / Split CSV files? *optional*	If **true**, the default, CSV or TSV files will be split. This means documents will be created for the unique rows found in the CSV file.

	- default - Adheres to the RFC4180 standard, but additionally allows empty lines to be skipped. - rfc - Adheres to the RFC4180 standard, which does not skip empty lines. - excel - A MS Excel format, using a comma as the delimiter. In some cases, the Excel locale determines a different delimiter, such as a ';'. Be sure to set the 'csvDelimterOverride' if your Excel application is configured to use a delimiter other than a comma. - mysql - The default MySQL format used by the SELECT INTO OUTFILE and LOAD DATA INFILE operations. This is a tab-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with '\'. The default is **default**.
csvWithHeader / Csv with Header? *optional*	If **true**, the first row of the CSV file will be parsed as a header and each row will be treated as column names, which will become field names for the values in each document. The default is **false**, which means that column names will be given numeric values as field names, starting with "0".
splitArchives / Split archive files? *optional*	If **true**, the default, .zip, .tar, .tar.gz, .tgz, .jar, .bzip, .bzip2, .cpio, and .dump files will be opened and documents found within the archive will be added to the index as individual documents. When archives are split, they are split recursively, meaning that multiple embedded archives will each be opened and indexed (e.g., if a .tar file contains a .zip file which contains a .csv file, the .csv file will be indexed and split into multiple documents according to the CSV-related properties). Note that .7z files are not supported at the current time.
csvDelimiterOverride / CSV delimiter-character override *optional*	Specify a column-delimiter character.
csvCommentOverride / CSV comment-character override *optional*	Specify the character used to indicate a comment row.
csvCharacterSetOverride / CSV character-set override *optional*	Specify the character set.

Other Configuration Properties

API Name / UI Label	Description
crawlDBType / Crawl-database type *optional*	The default value is "'in-memory". The other legal value is "on-disk". Crawl-database type "in-memory" uses a RAMStore-based crawldb during the crawl. At the end of the crawl, it writes the crawldb to disk as a binary compressed file whose

	named "data" and "data.p" written to the above directory throughout the crawl.
aliasExpiration / Alias expiration *optional*	The number of crawls after which an alias will expire. The default is **1** crawl.
retainOutlinks / Retain outlinks? *optional*	Default value is true. When true, the entire set of links that every single item links to is retained and stored in the crawldb. Enabling retainOutlinks and indexCrawlDBToSolr together will give you a copy of the links from each item as part of the Solr document, which can be useful for diagnostic purposes. Setting this property to false will lead to smaller crawldbs persisted on disk (in the case of both crawlDBType=in-memory and crawlDBType=on-disk), and in the case of crawlDBType=in-memory, less memory will be consumed during the crawl itself too. crawlDBType=in-memory means that the crawldb lives in memory for the entire crawl and is only persisted to disk at the end, so not retaining the entire set of links for every item saves a lot of RAM. This property will make a big difference in memory and disk consumption for web-crawls, where the vast majority of space occupied by each item in the crawldb is taken up by its links, usually. The crawldb shrunk by a factor of 10:1 with retainOutlinks=false for some web-crawls. It will make a minimal difference in filesystem crawls, where only directories have any links at all.
reevaluateCrawlDbOnStart / Reevaluate crawldb on start? *optional*	Default value is false. If true, on startup, Anda will check crawlDb and remove all illegal links from the crawlDb. Used when link-legality rules have been changed to cull set of links stored in crawlDb.
failFastOnStartLinkFailure/ Fail fast on start-link failure(s)? *optional*	Default value is true. If true, a first-time crawl fails as soon as a missing-start link is detected. It is difficult to figure out why many pages are missing after-the-fact, given a set of start links, each of which leads to swaths of pages. For a first-time crawl, it is reasonable to expect that all start links are valid, therefore, this property is true by default.
rewriteLinkScript / Link re-writing script *optional*	Specifies a JavaScript to perform link rewriting. Changing this field after crawling the content will require you to clear the crawldb.
restrictToTreeAllowSubdomains / Allow sub-domains in restrictToTree? *optional*	If **true**, this will allow links from any sub-domain of a URI in the startURIs list to pass link-legality checks. The default is **false**. Changing this field after crawling the content will require you to clear the crawldb.
restrictToTreeUseHostAndPath / Use paths in restrictToTree? *optional*	If **true**, the paths provided in URIs within the startLinks list will be used as part of link-legality checks. The default is **false**. Use this if you only want pages under the defined path(s) to be crawled instead of all documents found in the http://host.domain tree. For example, if you define "http://www.cnn.com/US/" as your startLink and only want to crawl URLs that start with that string, choose this option.

<table>
<tr><td>/ ignored host prefixes <em>optional</em></td><td>example, adding 'www.' to this list will allow URIs that have a valid host, but would otherwise be ignored because of the presence of the 'www.' prefix.
Changing this field after crawling the content will require you to clear the crawldb.</td></tr>
<tr><td>legalURISchemes / Legal URI schemes <em>optional</em></td><td>A list of URI schemes that are considered legal URIs for the crawl. This is expressed as a list in the REST API. The default is a list containing only '*', which makes all schemes legal.</td></tr>
<tr><td>retryEmit / Retry emitting? <em>optional</em></td><td>If <strong>true</strong>, the default, when a batch emit fails, documents will be tried one-by-one.</td></tr>
<tr><td>reevaluateCrawlDbOnStart / Reevaluate crawldb on start? <em>optional</em></td><td>If <strong>true</strong>, existing crawl database entries will be evaluated for legality at the start of the crawl. This allows for changing link legality rules (legalURISchemes) between crawls and then purging the crawl database of newly prohibited items.
The default is <strong>false</strong>.</td></tr>
<tr><td>collection / Collection <em>optional</em></td><td>The name of the document collection that documents will be indexed into.</td></tr>
<tr><td>initial_mapping / Initial field mapping <em>optional</em></td><td>A JSON map that applies a set of field mappings specific to a datasource which is applied before documents are sent to the index pipeline. The index pipeline may also include an additional field mapping stage. This could be useful if a single field mapping stage is used with multiple data sources; in this case, the initial_mapping property could be used to prepare incoming documents for the index pipeline stage.
When using the API, the JSON map should look the same as a field-mapping index stage, such as:<pre>
"initial_mapping": {
   "mappings": [
      {"source":"","target":"","operation":""},
      {"source":"","target":"","operation":""}
   ]
}
</pre>Copy
The crawler provides a default initial mapping for 'web' type crawls.</td></tr>
<tr><td>db / Connector DB <em>optional</em></td><td>Allows overriding the default ConnectorDb implementation. If it is not defined, the default will be used, which is defined in <code>{fusion_path}/connectors/plugins/<plugin>/connectors.json</code> . In most cases changing this property will not be required. If however, you find you need to change this, you can define a new ConnectorDb with the following additional properties. * <strong>type</strong>: a fully qualified class name of a subclass of <code>ConnectorDb</code> . If missing, the default is set to <code>com.lucidworks.connectors.db.impl.MapDbConnectorDb</code> which is a MapDb-based</td></tr>
</table>

	default. * **inv_aliases**: If true, the database will process and maintain a list of inverted aliases. This can be costly to performance, so this is set to false by default.

Property indexCrawlDBToSolr - index most recent crawldb in Solr

The boolean property indexCrawlDBToSolr when **true**, creates a Solr collection called 'crawldb_<datasource-ID>' which holds the crawldb for the *most recently completed crawl*. The default value is **false**.

The crawl must *finish*. Nothing is recorded if a crawl is stopped. Restricting the contents of the Solr collection to the most recently completed crawl limits the collection from growing very large over time. It means that at the point where a datasource is used to recrawl a website or filesystem, all information about previous crawls is deleted.

The resulting Solr documents have the following fields:

- id - the Solr uniqueKey field. The value is the concatenation of the map/table in the crawldb to which the doc belongs(see below), and the document ID. The two parts of the composite ID are separated by a '|' (gate/pipe) character. For example, the id of a document representing a FINISHED_MAP entry for a web-page in a web-crawl would look like: FINISHED_MAP|http://lucidworks.com/
- crawlCycle_ti - the crawl iteration, e.g., 1 for the initial crawl, 2 for the first re-crawl, etc.
- map_s - the map to which a document belongs in the crawldb

There are 6 kinds of information recorded:

- ALIAS_MAP
- INVERSE_ALIAS_MAP
- FINISHED_MAP
- ERRORS_MAP
- SIGNATURES_MAP
- DELETED_MAP

ALIAS_MAP is a mapping of all aliases (e.g. redirects in a web-crawl, symlinks in a filesystem crawl, etc.) to their canonical targets. INVERSE_ALIAS_MAP is the opposite: a mapping of canonicals to all of their aliases. FINISHED_MAP is all items that have been successfully indexed. ERRORS_MAP is all errors. SIGNATURES_MAP will only be there if dedupe is enabled, and it is a mapping of long signature hashes to their canonical item-ID. DELETED_MAP is all of the docs that were deleted in Solr in the last crawl, e.g. 404s that have failed enough times to be deleted in a web-crawl.

- parentID_s
- depth_ti
- fetchedDate_tdt
- emittedDate_tdt
- lastModified_tdt
- contentSignature_s
- discardMessage_s
- links_ss

Querying a crawldb Solr index

To see all errors with the exception that caused the error:

```
/solr/crawldb_mywebcrawl/select?q=map_s:ERRORS_MAP
```

Copy

To see all deleted items with any exception that lead to deleting them:

```
/solr/crawldb_mywebcrawl/select?q=map_s:DELETED_MAP
```

Copy

To see all items discovered via links on a particular page:

```
/solr/crawldb_mywebcrawl/select?q=parentID_s:<some ID>
```

Copy

To see all aliases of a particular page:

```
/solr/crawldb_mywebcrawl/select?q=id:INVERSE_ALIAS_MAP|<some ID>
```

Copy

Find all pages fetched in the last 24 hours: