# Fetchers

Fetchers are the implementation of the search using online services. Some fetchers require API keys to get them working. To get the fetchers running in a JabRef development setup, the keys need to be placed in the respective environment variable. The following table lists the respective fetchers, where to get the key from and the environment variable where the key has to be placed.

| Service | Key Source | Environment Variable | R |
|---|---|---|---|
| IEEEXplore | IEEE Xplore API portal | `IEEEAPIKey` | 20 |
| MathSciNet | (none) | (none) | De on cu ne |
| SAO/NASA Astrophysics Data System | ADS UI | `AstrophysicsDataSystemAPIKey` | 50 ca |
| ScienceDirect | | `ScienceDirectApiKey` | |
| SemanticScholar | https://www.semanticscholar.org/product/api#api-key-form | `SemanticScholarApiKey` | |
| Springer Nature | Springer Nature API Portal | `SpringerNatureAPIKey` | 50 ca |
| Zentralblatt Math | (none) | (none) | De on cu ne |
| Biodiversity Heritage Library | Biodiversitylibrary | `BiodiversityHeritageApiKey` | - |

“Depending on the current network” means that it depends on whether your request is routed through a network having paid access. For instance, some universities have subscriptions to MathSciNet.

On Windows, you have to log off and log on to let IntelliJ know about the environment variable change. Execute the gradle task `processResources` in the group “others” within IntelliJ to ensure

the values have been correctly written. Now, the fetcher tests should run without issues.

JabRef supports different kinds of fetchers:

- `EntryBasedFetcher`: Completes an existing bibliographic entry with information retrieved by the fetcher
- `FulltextFetcher`: Searches for a PDF for an exiting bibliography entry
- `SearchBasedFetcher`: Searches providers using a given query and returns a set of (new) bibliography entry. The user-facing side is implemented in the UI described at https://docs.jabref.org/collect/import-using-online-bibliographic-database.

There are more fetchers supported by JabRef. Investigate the package `org.jabref.logic.importer`. Another possibility is to investigate the inheritance relation of `WebFetcher` (Ctrl+H in IntelliJ).

## Fulltext Fetchers

- all fulltext fetchers run in parallel
- the result with the highest priority wins
- `InterruptedException` `ExecutionException` `CancellationException` are ignored

### Trust Levels

- `SOURCE` (highest): definitive URL for a particular paper
- `PUBLISHER`: any publisher library
- `PREPRINT`: any preprint library that might include non final publications of a paper
- `META_SEARCH`: meta search engines
- `UNKNOWN` (lowest): anything else not fitting the above categories

### Current trust levels

All fetchers are contained in the package `org.jabref.logic.importer.fetcher`. Here we list the trust levels of some of them:

- DOI: SOURCE, as the DOI is always forwarded to the correct publisher page for the paper
- ScienceDirect: Publisher
- Springer: Publisher
- ACS: Publisher
- IEEE: Publisher
- Google Scholar: META_SEARCH, because it is a search engine
- Arxiv: PREPRINT, because preprints are published there
- OpenAccessDOI: META_SEARCH

Reasoning:

- A DOI uniquely identifies a paper. Per definition, a DOI leads to the right paper. Everything else is good guessing.
- We assume the DOI resolution surely points to the correct paper and that publisher fetches may have errors: For instance, a title of a paper may lead to different publications of it. One the conference version, the other the journal version. –> the PDF could be chosen randomly

Code was first introduced at PR#3882.

## Background on embedding the keys in JabRef

The keys are placed into the `build.properties` file.

```
springerNatureAPIKey=${springerNatureAPIKey}
```

In `build.gradle`, these variables are filled:

```
"springerNatureAPIKey" : System.getenv('SpringerNatureAPIKey')
```

The `BuildInfo` class reads from that file and the key needs to be put into the map of default API keys in `JabRefCliPreferences::getDefaultFetcherKeys`.

```
keys.put(SpringerFetcher.FETCHER_NAME, buildInfo.springerNatureAPIKey);
```

The fetcher api key can then be obtained by calling the preferences.

```
importerPreferences.getApiKey(SpringerFetcher.FETCHER_NAME);
```

When executing `./gradlew run`, gradle executes `processResources` and populates `build/build.properties` accordingly. However, when working directly in the IDE, Eclipse keeps reading `build.properties` from `src/main/resources`. In IntelliJ, the task `JabRef Main` is executing `./gradlew processResources` before running JabRef from the IDE to ensure the `build.properties` is properly populated.

## Committing and pushing changes to fetcher files

Fetcher tests are run when a PR contains changes touching any file in the `src/main/java/org/jabref/logic/importer/fetcher/` directory. Since these tests rely on remote services, some of them may fail due to the network or the external server.

To learn more about doing fetcher tests locally, see Fetchers in tests in Testing.