# Table of Contents

Introduction .................................................................................................................. 1
Requirements .................................................................................................................. 2
Data format and schema changes ..................................................................................... 3
  Indexes .......................................................................................................................... 3
  Outbox polling system tables ....................................................................................... 3
Configuration changes ...................................................................................................... 5
API changes ..................................................................................................................... 6
SPI changes ...................................................................................................................... 7
Behavior changes ............................................................................................................ 8
Introduction

The aim of this guide is to assist you migrating an existing application using any version 6.2.x of Hibernate Search to the latest of the 7.0.x series.

If you think something is missing or something does not work, please contact us.

If you're looking to migrate from an earlier version, you should migrate step-by-step, from one minor version to the next, following the migration guide of each version.

To Hibernate Search 5 users

Be aware that a lot of APIs have changed since Hibernate Search 5, some only because of a package change, others because of more fundamental changes (like moving away from using Lucene types in Hibernate Search APIs).

When migrating from Hibernate Search 5, you are encouraged to migrate first to Hibernate Search 6.0 using the 6.0 migration guide, and only then to later versions (which will be significantly easier).
Requirements

Hibernate Search 7.0.0.Beta1 now:

• is using JDK 11 as the baseline.
• is based on Lucene 9 for its Lucene backend.
• requires using Hibernate ORM versions from the 6.2.x family.

When migrating from previous versions of Hibernate Search

• If you were using Hibernate ORM 6.2 and your corresponding Hibernate Search dependencies included references to artifacts ending with -orm6 (e.g. hibernate-search-mapper-orm-orm6) just remove the -orm6 suffix (e.g. use hibernate-search-mapper-orm instead) when you upgrade to this version of Hibernate Search.

• If you were using previous Hibernate ORM versions (e.g. Hibernate ORM 5.6), you would need to migrate to Hibernate ORM 6.2 first before upgrading to this version of Hibernate Search. See Hibernate ORM migration guides.
Data format and schema changes

Indexes

Elasticsearch indexes created with Hibernate Search 6.2 can be read from and written to with Hibernate Search 7.0.0.Beta1.

Reading and writing to Lucene indexes created with Hibernate Search 6.2 using Hibernate Search 7.0.0.Beta1 may lead to exceptions, since there were incompatible changes applied to internal fields. You must recreate your Lucene indexes and reindex your database. The easiest way to do so is to use the MassIndexer with dropAndCreateSchemaOnStart(true).

Outbox polling system tables

If you use the incubating outbox-polling coordination strategy, with the PostgreSQL database you will be impacted by the changes to entities that represents the outbox event and agent, requiring database schema changes. The payload column changes its type from oid to bytea for both outbox event and agent tables. You can find suggested migration scripts below:

Postgresql:

```sql
-- Change outbox event `payload` column type to bytea:
-- Note the way existing LOBs are retrieved using lo_get function.
alter table hsearch_outbox_event
  alter column payload type bytea using lo_get(payload);

-- Change agent `payload` column type to bytea:
-- Note: even though agent table should've not used payload column so far, we still need to have a type cast.
-- It can be done either with the same `lo_get()` function call, or with a pair of casts like `cast( cast( payload as text ) as bytea )`:
alter table hsearch_agent
  alter column payload type bytea using lo_get(payload);
```

Other databases:

- CockroachDB: no migration required. Type of the payload is bytes in both cases.
- MySQL: no migration required. Type of the payload is longblob in both cases.
- MariaDB: no migration required. Type of the payload is longblob in both cases.
- DB2: no migration required. Type of the payload is blob in both cases.
- Oracle: no migration required. Type of the payload is blob in both cases.
- MSSQL: no migration required. Type of the payload is varbinary(max) in both cases.
- H2: no migration required. Type of the payload is blob in both cases.

In case database migration cannot be performed immediately when upgrading to a new version of Hibernate Search, a pair of configuration properties is available:

```properties
hibernate.search.coordination.entity.mapping.agent.payload_type=materialized_blob
```
Keep in mind that these properties are temporary, to help with the migration, and will be removed in the future versions of Hibernate Search.
Configuration changes

The configuration properties are backward-compatible with Hibernate Search 6.2.
API changes

The API is backward-compatible with Hibernate Search 6.2.
SPI changes

The SPI are backward-compatible with Hibernate Search 6.2.
Behavior changes

The default value for `hibernate.search.backend.query.shard_failure.ignore` is changed from `true` to `false` which means that now Hibernate Search will throw an exception if at least one shard failed during a search operation. To get the previous behavior set this configuration property explicitly to `true`. Note, this setting must be set for each elasticsearch backend, if multiple are defined.