PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide

Contents

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Advancing Controller setup guide

This guide provides step-by-step instructions to help you configure and deploy the Advancing Controller effectively.

Infrastructure prerequisites

Advancing controller requires the following components to be set up before it can be started:

- **FLOWX.AI Engine deployment** - the Advancing Controller is dependent on the FLOWX.AI Engine and must be deployed in the same environment, refer to the FLOWX.AI Engine setup guide for more information on how to set up the Engine
- **DB instance** - the Advancing Controller uses a PostgreSQL or OracleDB as database instance

Dependencies

- Database
- Datasource
- FLOWX.AI Engine

Database configuration

Postgres

A basic Postgres configuration for Advancing:

```yaml
postgresql:
  enabled: true
  postgresqlUsername: "postgres"
  postgresqlPassword: ""
  postgresqlDatabase: "advancing"
  existingSecret: "postgresql-generic"
  postgresqlMaxConnections: 200
  persistence:
    enabled: true
    storageClass: standard-rwo
    size: 20Gi
  resources:
    limits:
      cpu: 1000m
      memory: 1024Mi
    requests:
      memory: 256Mi
      cpu: 100m
  metrics:
    enabled: true
    serviceMonitor:
      enabled: false
    prometheusRule:
      enabled: false
  primary:
    nodeSelector:
      preemptible: "false"
```

CAUTION

If the parallel advancing configuration already exists, resetting the 'advancing' database must be done by executing the SQL command `DROP DATABASE advancing;`. Once the database has been dropped, the Liquibase script will automatically re-enable it.

Configuration

The following configuration details need to be added using environment variables:

Advancing controller uses a PostgreSQL or an Oracle database as a dependency.

- the user, password, connection link, and database name need to be configured correctly, if these details are not configured correctly, errors will occur at startup
- the datasource is configured automatically via a Liquibase script inside the engine. All updates will include migration scripts.

Configuring datasource

The following configuration details need to be added using environment variables:

- `SPRING_DATASOURCE_URL` - environment variable used to configure a data source URL for a Spring application, it typically contains the JDBC driver name, the server name, port number, and database name

- `SPRING_DATASOURCE_USERNAME` - environment variable used to set the username for the database connection, this can be used to connect to a database instance

- `SPRING_DATASOURCE_PASSWORD` - environment variable used to store the password for the database connection, this can be used to secure access to the database and ensure that only authorized users have access to the data

- `SPRING_JPA_DATABASE` - relevant because it is used to specify the type of database that the Spring application should connect to (accepted values:

`oracle` or `postgresql`)

- `SPRING_JPA_PROPERTIES_HIBERNATE_DEFAULTSCHEMA` (❗ only for Oracle DBs) - dpecifies the default schema to use for the database (default value: `public`)

You will need to make sure that the user, password, connection link and db name are configured correctly, otherwise, you will receive errors at start time.

> ⚠ **CAUTION**
>
> It's important to keep in mind that the Advancing Controller is tightly integrated with the FLOWX.AI Engine. Therefore, it is important to ensure that both the Engine and the Advancing Controller are configured correctly and are in sync.

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access rights for Engine

Granular access rights can be configured for restricting access to the Engine component.

Two different access authorizations are provided, each with specified access scopes:

1. **Manage-processes** - for configuring access for running test processes

Available scopes:

- **edit** - users are able to start processes for testing and to test action rules

2. **Manage-instances** - for configuring access for manipulating process instances

Available scopes:

- **read** - users can view the list of process instances
- **admin -** users are able to retry an action on a process instance token

The Engine service is preconfigured with the following default users roles for each of the access scopes mentioned above:

- **manage-processes**
    - edit:
        - ROLE_ADMIN_MANAGE_PROCESS_EDIT
        - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
    - admin:
        - ROLE_ADMIN_MANAGE_PROCESS_ADMIN
- **manage-instances**
    - read:

        - ROLE_ENGINE_MANAGE_INSTANCE_READ
        - ROLE_ENGINE_MANAGE_INSTANCE_ADMIN
    - admin:
        - ROLE_ENGINE_MANAGE_INSTANCE_ADMIN

DANGER

These roles need to be defined in the chosen identity provider solution.

In case other custom roles are needed, you can configure them using environment variables. More than one role can be set for each access scope.

To configure access for each of the roles above, adapt the following input:

`SECURITY_ACCESSAUTHORIZATIONS_AUTHORIZATIONNAME_SCOPES_SCOPENAME_ROLESALLOWED:NEEDED_ROLE_NAMES`

Possible values for AUTHORIZATIONNAME: MANAGEPROCESSES, MANAGEINSTANCES.

Possible values for SCOPENAME: read, edit, admin.

For example, if you need to configure role access for read, insert this:

```
SECURITY_ACCESSAUTHORIZATIONS_MANAGEINSTANCES_SCOPES_READ_ROLE
ROLE_NAME_TEST
```

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Configuring access roles for processes

Access to a process definition

Setting up user role-based access on process definitions is done by configuring swimlanes on the process definition.

» Swimlanes

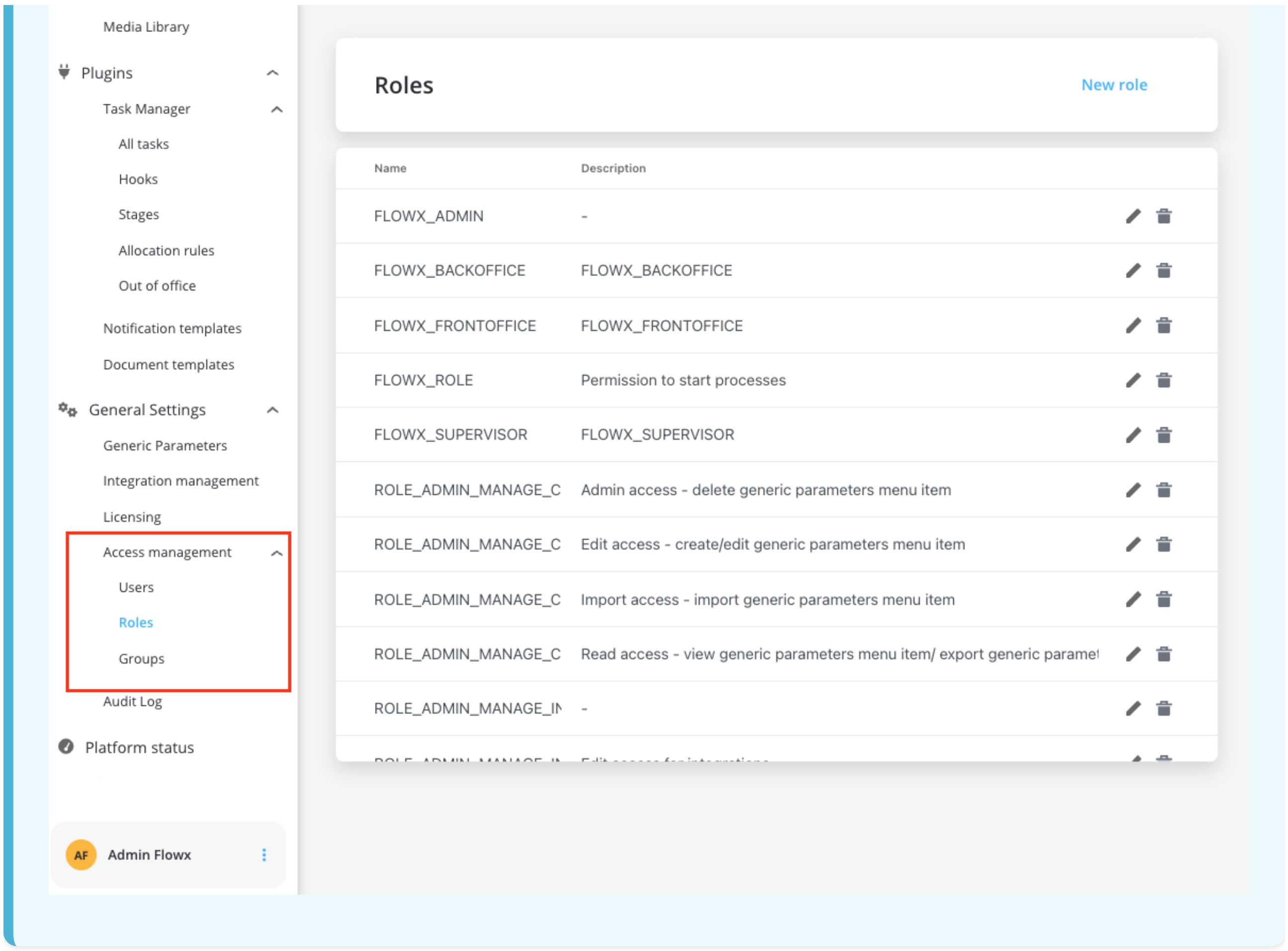
By default, all process nodes belong to the same swimlane. If more swimlanes are needed, they can be edited in the process definition settings panel.

Swimlane role settings apply to the whole process, the process nodes or the actions to be performed on the nodes.

First, the desired user roles need to be configured in the identity provider solution and users must be assigned the correct roles.

INFO

You can use the **Access management** tab under **General Settings** to administrate all the roles.



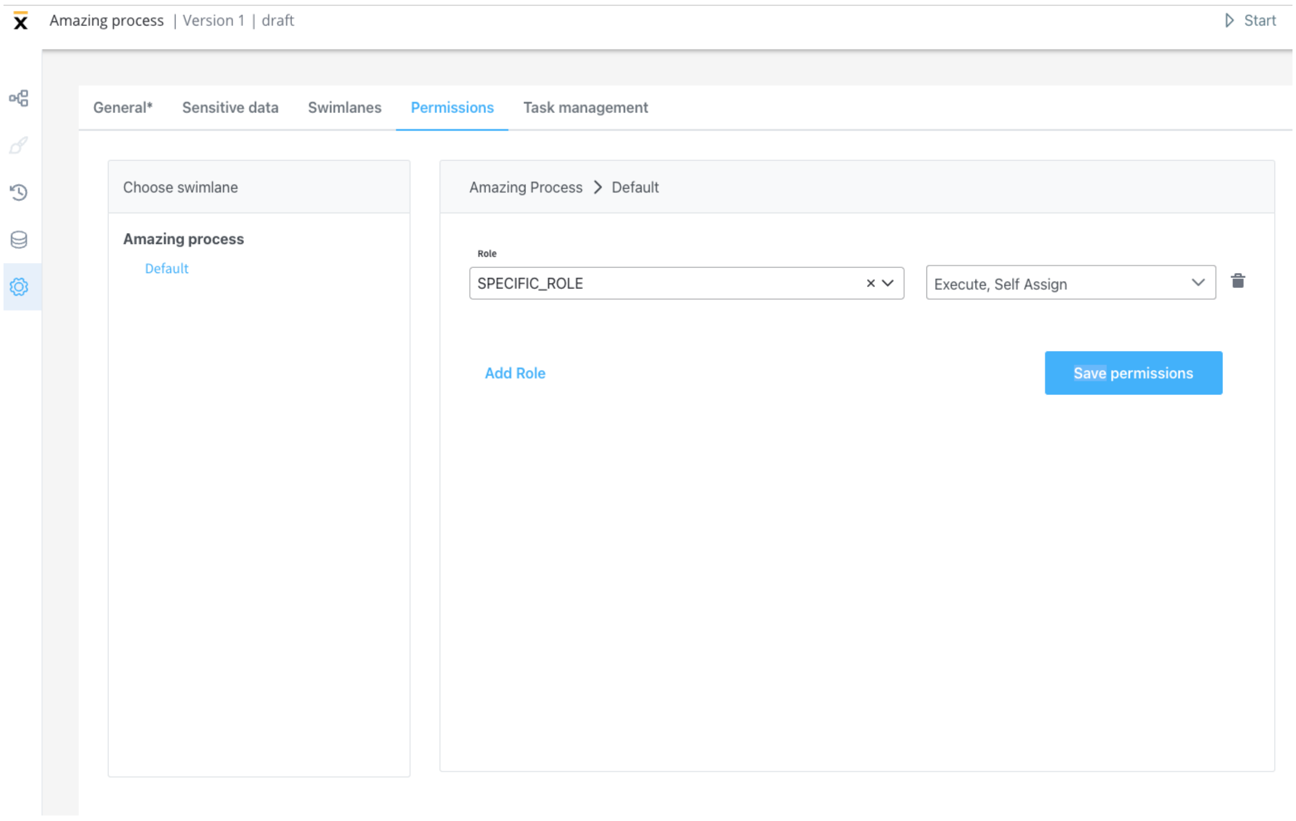
Name	Description
FLOWX_ADMIN	-
FLOWX_BACKOFFICE	FLOWX_BACKOFFICE
FLOWX_FRONTOFFICE	FLOWX_FRONTOFFICE
FLOWX_ROLE	Permission to start processes
FLOWX_SUPERVISOR	FLOWX_SUPERVISOR
ROLE_ADMIN_MANAGE_C	Admin access - delete generic parameters menu item
ROLE_ADMIN_MANAGE_C	Edit access - create/edit generic parameters menu item
ROLE_ADMIN_MANAGE_C	Import access - import generic parameters menu item
ROLE_ADMIN_MANAGE_C	Read access - view generic parameters menu item/ export generic paramet
ROLE_ADMIN_MANAGE_IN	-

DANGER

To be able to access the roles defined in the identity provider solution, a **service account** with appropriate permissions needs to be added in the identity provider. And the details of that service account **need to be set up in the platform configuration**.

The defined roles will then be available to be used in the process definition settings (**Permissions** tab) panel for configuring swimlane access.

A **Default** swimlane comes with two default permissions assigned based on a specific role.



Role: SPECIFIC_ROLE — Execute, Self Assign

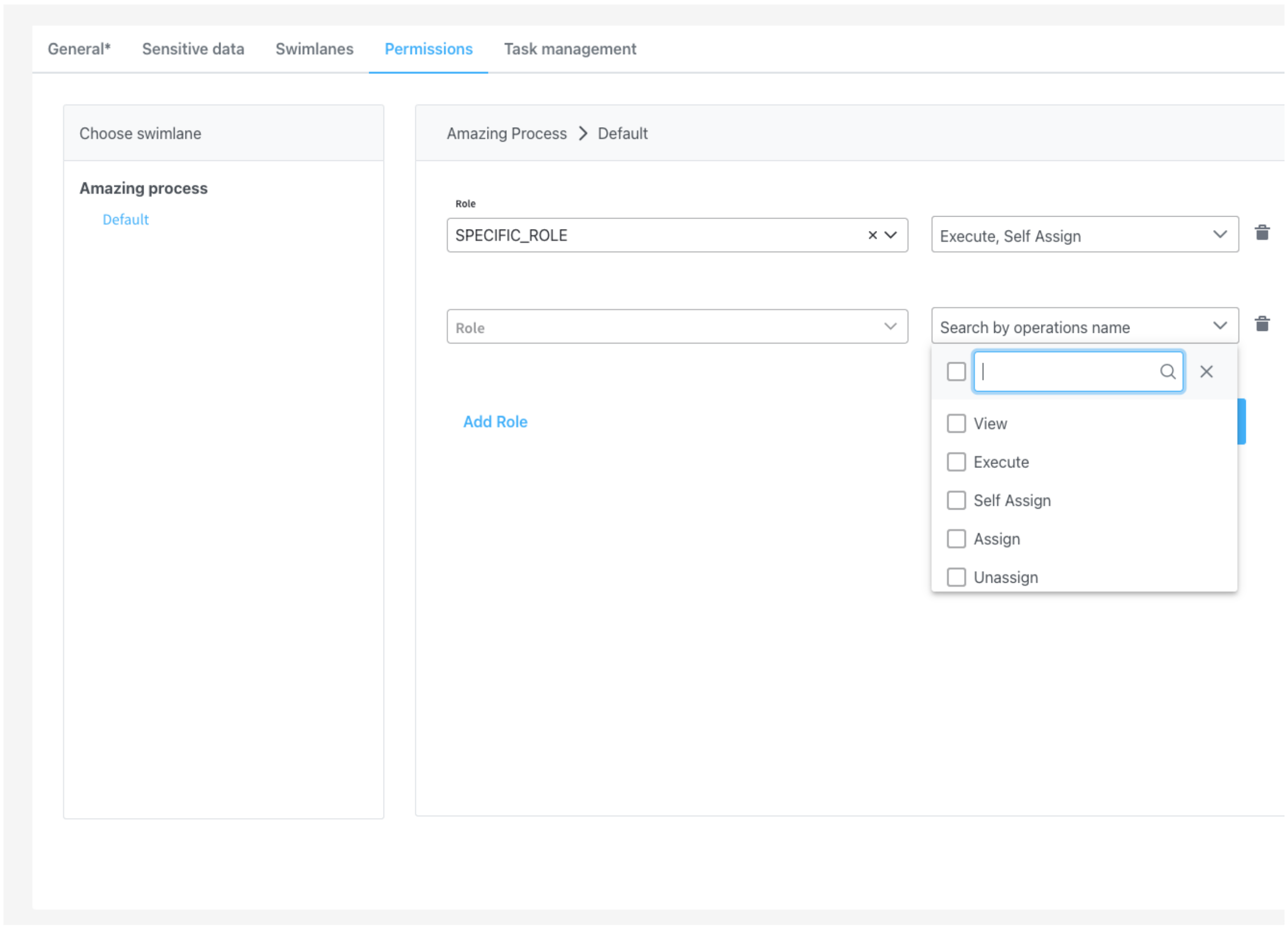
- **execute** - the user will be able to start process instances and run actions on them
- **self-assign** - the user can assign a process instance to them and start working on it

DANGER

This is valid for **> 2.11.0** FLOWX.AI platform release.

Other **Permissions** can be added manually, depending on the needs of the user. Some permissions are needed to be configured so you can use features inside Task Management plugin. Specific roles need to be assigned separately on a few available process operations. These are:

- **view** - the user will be able to view process instance data
- **assign** - user can assign tasks to other users (this operation is only accessible through the **Task management** plugin)
- **unassign** - user can unassign tasks from other users (this operation is only accessible through the **Task management** plugin)
- **hold** - user can mark the process instance as on hold (this operation is only accessible through the **Task management** plugin)
- **unhold** - user can mark the process instance as not on hold (this operation is only accessible through the **Task management** plugin)



DANGER

< 2.11.0 platform release - if no role is configured on an operation, no restrictions will be applied.

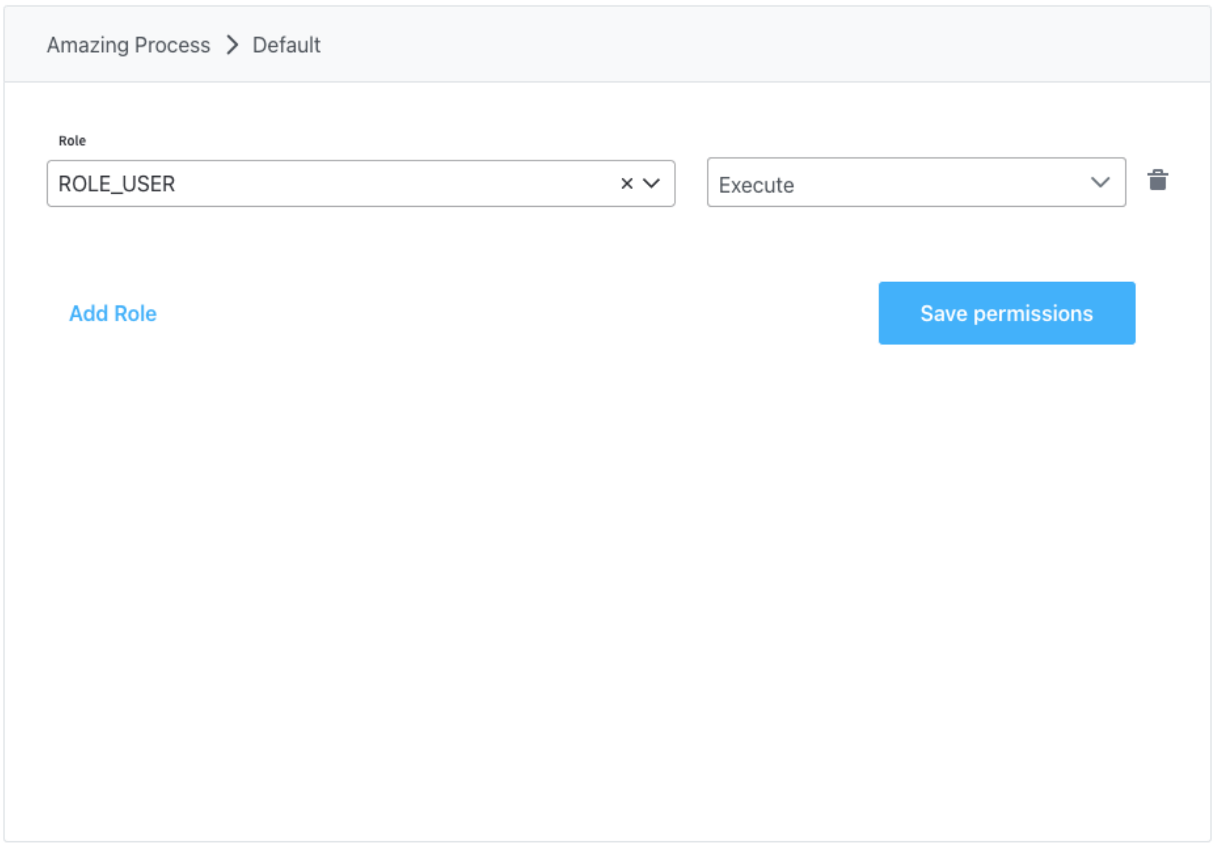
Configuration examples

CAUTION

Valid for < 2.11.0 release version.

Regular user

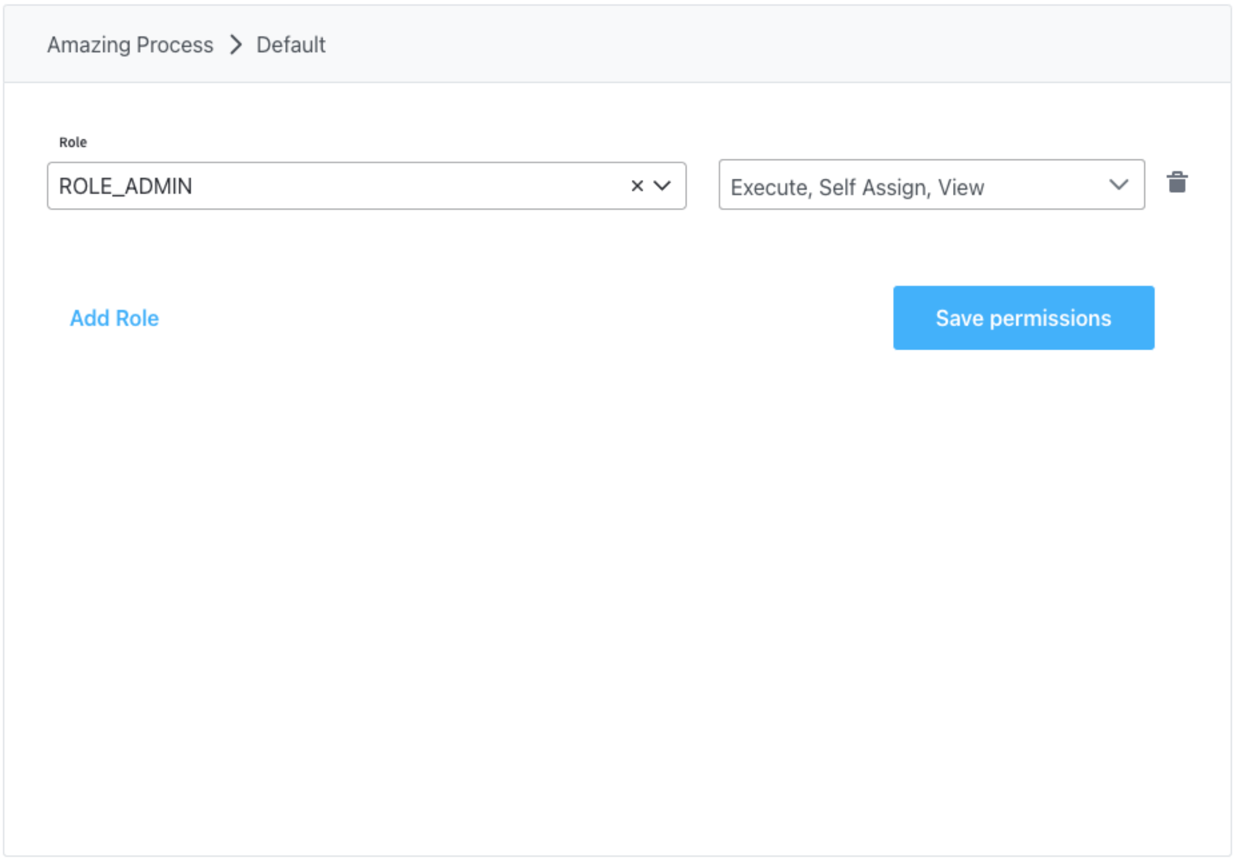
Below you can find an example of configuration of roles for a regular user:


Amazing Process > Default

Role: ROLE_USER — Execute

Admin

Below you can find an example of configuration of roles for an admin user:



CAUTION

❗ Starting with **2.11.0** release, specific roles are needed, otherwise, restrictions will be applied.

After setting up your preferred identity provider solution, you will need to add the desired access roles in the application configuration for the FLOWX Engine (using environment variables):

» Authorization & access roles

Restricting process instance access based on business filters

» Business filters

Before they can be used in the process definition the business filter attributes need to be set in the identity management platform. They have to be configured as a list of filters and should be made available on the authorization token. Application users will also have to be assigned this value.

Viewing processes instances

Active process instances and their related data can be viewed from the FLOWX Designer. A user needs to be assigned to a specific role in the identity provider solution to be able to view this information.

By default, this role is named `FLOWX_ROLE`, but its name can be changed from the application configuration of the Engine by setting the following environment variable:

`FLOWX_PROCESS_DEFAULTROLES`

When viewing process instance-related data, it can be configured whether to hide specific sensitive user data. This can be configured using the `FLOWX_DATA_ANONYMIZATION` environment variable.

Access to REST API

To restrict API calls by user role, you will need to add the user roles in the application config:

```
security:
  pathAuthorizations:
    -
      path: "/api/**"
      rolesAllowed: "ANY_AUTHENTICATED_USER" or
"USER_ROLE_FROM_IDENTITY_PROVIDER"
```

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Process instance indexing / Configuration guidelines

The configuration of Elasticsearch for process instances indexing depends on various factors related to the application load, the number of process instances, parallel requests, and indexed keys per process. Although the best approach to sizing and configuring Elasticsearch is through testing and monitoring under load, here are some guidelines to help you get started:

Indexing strategy

When deleting data in Elasticsearch, it's recommended to delete entire indices instead of individual documents. Creating multiple smaller indices provides the flexibility to delete entire indices of old data that are no longer needed.

- Advantages of Multiple Small Indices:
    - Fast indexing process.
    - Flexibility in cleaning up old data.
- Potential Drawbacks:
    - Hitting the maximum number of shards per node, resulting in exceptions when creating new indices.
    - Increased search response time and memory footprint.

Alternatively, you can create fewer indices that span longer periods of time, such as one index per year. This approach offers small search response times but may result in longer indexing times and difficulty in cleaning up and recovering data in case of failure.

» What is indexing?

Shard and replica configuration

The solution includes an index template that gets created with the settings from the process-engine app (name, shards, replicas) when running the app for the first time. This template controls the settings and mapping of all newly created indices.

» What is sharding?

» Index template

Once an index is created, you cannot update its number of shards and replicas. However, you can update the settings from the index template at runtime in Elasticsearch, and new indices will be created with the updated settings. Note that the mapping should not be altered as it is required by the application.

Recommendations for resource management

To manage functional indexing operations and resources efficiently, consider the following recommendations:

- Sizing indexes upon creation
- Balancing
- Delete unneeded indices
- Reindex large indices
- Force merge indices
- Shrink indices
- Combine indices

Sizing indexes upon creation

Recommendations:

- Start with monthly indexes that have 2 shards and 1 replica. This setup is typically sufficient for handling up to 200k process instances per day; ensures a parallel indexing in two main shards and has also 1 replica per each main shard (4 shards in total). This would create 48 shards per year in the elastic search nodes; A lot less than the default 1000 shards, so you will have enough space for other indexes as well.

  - If you observe that the indexing gets really, really slow, then you should look at the physical resources / shard size and start adapting the config.
  - If you observe that indexing one monthly index gets massive and affects the performance, then think about switching to weekly indices.
  - If you have huge spikes of parallel indexing load (even though that depends on the Kafka connect cluster configuration), then think about adding more main shards.

- Consider having at least one replica for high availability. However, keep in mind that the number of replicas is applied to each shard, so creating many replicas may lead to increased resource usage.
- Monitor the number of shards created and estimate when you might reach the maximum shards per node, taking into account the number of nodes in your cluster.

Balancing

When configuring index settings, consider the number of nodes in your cluster. The total number of shards (calculated by the formula: primary_shards_number * (replicas_number +1)) for an index should be directly proportional to the number of nodes. This helps Elasticsearch distribute the load evenly across nodes and avoid overloading a single node. Avoid adding shards and replicas unnecessarily.

Delete unneeded indices

Deleting unnecessary indices reduces memory footprint, the number of used shards, and search time.

Reindex large indices

If you have large indices, consider reindexing them. Process instance indexing involves multiple updates on an initially indexed process instance, resulting in multiple versions of the same document in the index. Reindexing creates a new index with only the latest version, reducing storage size, memory footprint, and search response time.

Force merge indices

If there are indices with no write operations performed anymore, perform force merge to reduce the number of segments in the index. This operation reduces memory footprint and response time. Only perform force merge during off-peak hours when the index is no longer used for writing.

Shrink indices

If you have indices with many shards, consider shrinking them using the shrink operation. This reindexes the data into an index with fewer shards. Perform this operation during off-peak hours.

Combine indices

If there are indices with no write operations performed anymore (e.g., process_instance indices older than 6 months), combine these indices into a larger one and delete the smaller ones. Use the reindexing operation during off-peak hours. Ensure that write operations are no longer needed from the FLOWX platform for these indices.

Was this page helpful?

PLATFORM SETUP GUIDES / FLOWX.AI Engine setup guide / Old access roles

CAUTION

Deprecated since platform version 1.16.0

Old access roles

Access to a process definition

You can restrict access to process definitions by user roles. This can be done by setting the desired operation permissions on a process definition.

Start by adding the needed roles in the database. These need to match the roles configured in the identity provider solution. Each role can have one or more permissions defined on it. Permissions can be applied to all users or only to the owner of the specific resource (for example the person that started the process instance).

After saving a new process definition, you can also save specific user roles for it to restrict user access. Access rights can be defined on the following operations that can be performed on a process definition:

- starting a new instance of the process definition
- viewing the instance of that process definition

Here's an example of setting operation permissions for a process definition:

```
{
    "START": ["PROCESS_START"],
    "VIEW": ["PROCESS_VIEW", "PROCESS_VIEW_ALL"]
}
```

where `START` and `VIEW` are the possible operations to be performed on the definitions and `PROCESS_START`, `PROCESS_VIEW`, `PROCESS_VIEW_ALL` are permissions stored in the database.

Access to actions from process definitions

Operation permissions can also be set on specific nodes in order to restrict the access to the actions defined on that node. This can be done similarly to setting operation permissions on process definitions. The operation name to be used for nodes is `NODE_RUN`.

As nodes also hold the definitions for the user interface, deciding which user role can see a certain UI template can also be done by using node permissions. The templates linked to a node can only be viewed by a user that has the `NODE_RUN` permission on that node, if the access on that node is restricted.

Access to a process definition

You can restrict access to process definitions by user roles. This can be done by setting the desired operation permissions on a process definition.

Start by adding the needed roles in the database. These need to match the roles configured in the identity provider solution. Each role can have one or more permissions defined on it. Permissions can be applied to all users or only to the owner of the specific resource (for example the person that started the process instance).

After saving a new process definition, you can also save specific user roles for it to restrict user access.

Access rights can be defined on the following operations that can be performed on a process definition:

- starting a new instance of the process definition
- viewing the instance of that process definition

Here's an example of setting operation permissions for a process definition:

```
{
    "START": ["PROCESS_START"],
    "VIEW": ["PROCESS_VIEW", "PROCESS_VIEW_ALL"]
}
```

where `START` and `VIEW` are the possible operations to be performed on the definitions and `PROCESS_START`, `PROCESS_VIEW`, `PROCESS_VIEW_ALL` are permissions stored in the database.

Access to actions from process definitions

Operation permissions can also be set on specific nodes in order to restrict the access to the actions defined on that node. This can be done similarly to setting operation permissions on process definitions. The operation name to be used for nodes is `NODE_RUN`.

As nodes also hold the definitions for the user interface, deciding which user role can see a certain UI template can also be done by using node permissions. The templates linked to a node can only be viewed by a user that has the `NODE_RUN` permission on that node, if the access on that node is restricted.

Was this page helpful?