Creatio Academy

Containerized components

Version 8.0

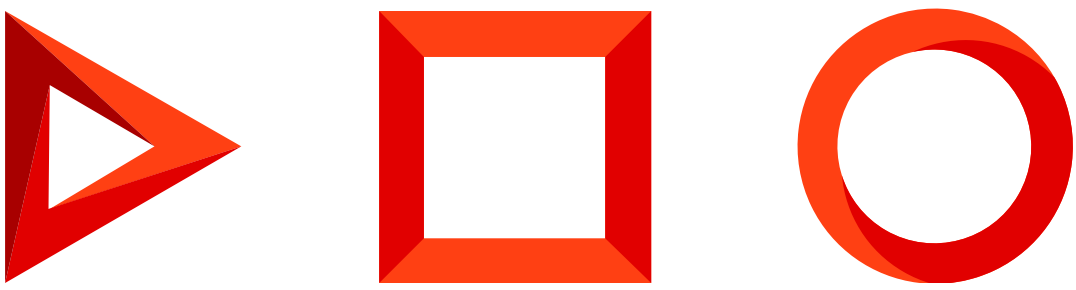


Table of Contents

Global search

PRODUCTS: ALL CREATIO PRODUCTS

The Global Search Service integrates ElasticSearch with Creatio. It performs the following functions:

- **Recording**:
  - Subscribes clients by creating an index in ElasticSearch and saves the connection between the index and the application.
  - Disconnects clients by removing their index in ElasticSearch.
- **Transporting**:
  - Participates in the indexing process by retrieving data from the database.

The sequence of actions during the global search setup depends on the global search version you are going to use. We recommend always using the latest version of the global search service with the latest version of Creatio.

You can deploy the components of global search version 3.0 using **Kubernetes orchestrator and Helm package manager** or **Docker**.

We also recommend backing up ElasticSearch daily to ensure the correct operation of the service and to enable the restoration of data after failures, e. g., power outages.

If you have any questions during the setup, we recommend consulting the Global search and deduplication FAQ.

Attention. You need repository access to set up the current version of the global search service. Contact Creatio support to verify your license and gain access to the repository.

Set up the global search service using Kubernetes

To set up the service, download the source files. Download files.

To install the service:

1. Set up the target environment:
   a. **Kubernetes cluster**. Learn more about setting up and managing the cluster in the Kubernetes documentation.
   b. **Helm package manager**. Learn more about installing the package manager in the Helm documentation.
2. Unpack the **values-onsite.yaml** file from the source file archive and save the file to the same directory as the archive.
3. Open the file. View the main parameters the file uses in the table below.
4. Specify the public service URL in the format of http://k8s-node:30332 in the `global.searchService.url` variable.

5. If you **have not installed** Redis, RabbitMQ, ElasticSearch, and PostgreSQL service database yet, proceed to the next step.

   If you **have already installed** these services, for example, when setting up other containerized components, disable the installation of the services and set up the connection to existing services in the **values-onsite.yaml** source file of the global search service.

   > **Note.** For highly loaded environments, we recommend deploying ElasticSearch in a cluster. Learn more in ElasticSearch documentation.

   - For **Redis**:
     - Disable the service installation. To do this, specify `enabled: false` in the `redis` section of the **values-onsite.yaml** file.

       ```
       redis:
         enabled: false
       ```

     - Set up the connection to the previously installed Redis. To do this, specify the connection parameters in the `global.redis` section of the **values-onsite.yaml** file.

       ```
       global:
         redis:
           host: [host]
           port: [port]
           database: [database]
       ```

       Where

       [host] is the address of the Redis server.

       [port] is the connection port of the Redis server.

       [database] is the name of the Redis database.

   - For **RabbitMQ**:
     - Disable the service installation. To do this, specify `enabled: false` in the `rabbitmq` section of the **values-onsite.yaml** file.

       ```
       rabbitmq:
         enabled: false
       ```

     - Set up the connection to the previously installed RabbitMQ. To do this, specify the connection parameters in the `global.rabbitmq` section of the **values-onsite.yaml** file.

```
global:
  rabbitmq:
    host: [host]
    vhost: [vhost]
    port: [port]
    user: [user]
    password: [password]
```

Where

[host] is the address of the RabbitMQ service.

[vhost] is the virtual host address of the RabbitMQ service.

[port] is the amqp connection port of RabbitMQ.

[user] is the RabbitMQ user.

[password] is the RabbitMQ user password.

- For **PostgreSQL** service database:
  - Disable the PostgreSQL installation. To do this, specify `enabled: false` in the `postgresql` section of the **values-onsite.yaml** file.

```
postgresql:
  enabled: false
```

  - Set up the connection to the PostgreSQL service database. To do this, specify the connection parameters in the `global.postgresql` section of the **values-onsite.yaml** file.

```
global:
  db:
    user: [user]
    password: [password]
    database: [database]
    host: [host]
    port: [port]
```

Where

[user] is the PostgreSQL user on whose behalf to connect to the database.

[password] is the PostgreSQL user password.

[database] is the PostgreSQL service database.

[host] is the address of the PostgreSQL database.

[port] is the port to connect to the database.

- For **ElasticSearch**:
  - Disable ElasticSearch installation. To do this, specify `enabled: false` in the `elasticsearch` section of the **values-onsite.yaml** file.

    ```
    elasticsearch:
      enabled: false
    ```

  - Set up the connection to the previously installed ElasticSearch. To do this, specify the connection parameters in the `global.elasticsearch` section of the **values-onsite.yaml** file.

    ```
    global:
      elasticsearch:
        url: [url]
        user: [user]
        password: [password]
    ```

    Where

    [user] is the ElasticSearch user.

    [password] is the ElasticSearch user password.

    [url] is the ElasticSearch service URL in the format of http://elasticsearch:9200.

6. Run the `helm install gs -f values-onsite.yaml globalsearch.tgz` command. As a result, Helm will install the global search service and selected dependencies.

> **Note.** By default, Helm deploys services with NodePort type.

The main parameters of the global search service the **values.yaml** file uses.

Parameter	Parameter description
scheduler.env.fillQueueInterval	The run interval for primary indexing, in milliseconds.
worker.env.indexingCommandTimeout	The timeout of the Creatio database query upon primary indexing, in seconds.
workerSingle.env.indexingCommandTimeout	The timeout of the Creatio database query upon instant indexing, in seconds.
global.incrementDays	The number of days within which to index the changed records during a single primary indexing iteration. Affects the indexing speed and Creatio database load. The higher the value, the faster the indexing and higher the load. The lower the value, the longer the indexing and lower the load.
log4Net	Logging settings.
global.indexingContentLength	The maximum length of text fields upon indexing.
global.elasticsearch	ElasticSearch connection parameters.
global.searchService	Public URL to the search-service in the format of http://k8s-node:30332.
global.rabbitmq	RabbitMQ connection settings.
global.db	The connection settings of the global search service's internal service database.
global.redis	Redis connection settings.

Set up the global search service in Docker

The global search requires 2 dedicated physical or virtual servers (“server 1” and “server 2”) with Linux installed. Use the requirements calculator to check the server requirements.

Attention. The global search setup procedure depends on the version you intend to install.

Learn more about the OS versions supported by Docker in the Docker documentation. Depending on your company needs, you can use either Docker Community Edition (CE) or Enterprise Edition (EE). Learn more in the Docker documentation.

Note. The procedure below is relevant for global search service version 3.0. If you need to set up an

earlier version of the global search, follow the procedure described in Creatio 7.16 documentation.

To **update** global search version 2.0 to version 3.0, run the docker-compose down -v command to delete all docker volumes on servers 1 and 2, then install and set up the services once again.

Global search components

Deploy on server 1:

- elasticsearch. The search engine.

Deploy on server 2:

- postgres. The database for configuring the global search components.
- rabbitmq. The message broker.
- redis. The database used for caching and performance improvement.
- gs-web-api. The web service that configures global search components.
- gs-web-indexing-service. The web service that processes requests to perform targeted indexing of Creatio data.
- gs-search-service. An elasticsearch proxy web service for data search.
- gs-scheduler. The scheduler of Creatio data indexing in ElasticSearch.
- gs-worker. The component that indexes Creatio data in ElasticSearch as per the scheduler tasks.
- gs-worker-replay. The component that processes the indexing results (gs-worker results).
- gs-worker-single. The component that performs the targeted indexing of business process data in ElasticSearch upon a request from the business process.
- gs-worker-single-replay. The component that processes exceptions as part of the targeted indexing (gs-worker-single results).
- gs-worker-single-task. The component that schedules tasks for gs-worker-single.
- gs-worker-queried-single-task. The component that generates tasks for gs-worker-single.

To set up the components, download the source files. Download files.

The list of ports used by global search components:

Attention. If you are using a firewall, make sure the ports below are open and available.

Component name	Outgoing port	Incoming port	Notes
gs-web-api		81	Configure the incoming port using the WEB_API_PORT variable
gs-web-indexing-service		82	Configure the incoming port using the WEB_INDEXING_SERVICE_PORT variable
gs-search-service	9200	83	Configure the incoming port using the SEARCH_SERVICE_PORT variable
gs-worker	9200		Requires connection to the server where elasticsearch is located.
gs-worker-single	9200		Requires connection to the server where elasticsearch is located.
elasticsearch		9200	

Global search setup procedure

1. Install Docker on a physical or virtual machine running Linux OS.
2. Install Docker-Compose.
3. Install ElasticSearch.
4. Set up the container variables.
5. Install and run the Global Search Service components.
6. Enable the global search functionality in Creatio.

Install Docker

Install Docker on Linux to deploy global search components. The installation is covered in the Docker documentation.

Run the **docker --version** command on a Linux machine to verify the installed Docker version.

Install Docker-Compose

The installation of Docker-Compose is covered in the Docker documentation.

Install ElasticSearch

Note. This guide covers the procedure for deploying ElasticSearch in Docker-Compose. You can also

deploy ElasticSearch as OS daemon, which does not involve installing Docker and Docker-Compose. Learn more about the setup procedure in the ElasticSearch documentation.

To install ElasticSearch:

1. Go to the ElasticSearch installation server (server 1) and open the /opt directory.
2. Download and unpack the archive with setup files to the directory. Download the archive.
3. Open the /opt/docker-compose/elasticsearch directory (where the components are located) and run the following command:

   ```
   docker-compose up -d
   ```

   The command might take up to several minutes to complete.
4. Make sure that the log files do not contain any errors after the command is complete. To do this, run the following command:

   ```
   docker logs es-01
   ```

Set up the container variables

Configure the global search component containers via the file that contains the environment variables. The variables are stored in the /opt/compose/services/.env file. Edit this file to set the variables.

Variable name	Details	Default value
GS_ES_URL	The external ElasticSearch host required for access from Creatio. Specify the IP address of the server where the ElasticSearch is deployed.	http://elasticsearch-publicip:9200
CURRENT_SERVER_IP	The external IP address of the server where the global search components are deployed (server 2).	10.0.0.1

Note. To check the external IP address of the server, run the `hostname -I | awk '{ print $1 }'` command.

Additional variables that control the data indexing parameters in ElasticSearch

Variable name	Details	Default value
GS_DB_INCREMENT_DAYS	Number of days to index per one scheduler iteration. ModifiedOn columns of Creatio records are used for comparison.	500 days
GS_DB_FILL_QUEUE_INTERVAL	Creatio database data collection interval for the regular scheduler. The lower the variable, the higher the load on Creatio database and the faster the primary indexing.	30000 (specified in milliseconds)

Run containers that have Global Search Service components

Attention. For the correct container operation, the UTC time on the Linux machine with Docker installed must correspond to the UTC time on the Creatio database server. The permissible deviation is up to five minutes. Otherwise, the global search might not index all records.

1. Go to the server that has the global search components (server 2) and open the /opt folder.
2. Download and unpack the archive with setup files to the opened folder. Download the archive.
3. Open the /opt/compose/services folder and run the following command:

```
docker-compose up -d
```

Verify that containers ran successfully

To see the running global search containers, use the following console command:

```
docker ps --filter "label=service=gs" -a --format "table {{.Names}}\t{{.Ports}}\t{{.Status}}\t{{
```

All currently running containers display the **Up** status.

Logging

By default, the container logging takes place in the stdout and stderr.

Note. The “docker logs --tail 100 gs-worker” command displays 100 last strings from the gs-worker container.

Note. The mysql or rabbitmq containers might become temporarily unavailable on start because they run after the rest of the containers. In this case, wait until a notification about the successful container connection and start appears in the log files: “Now listening on: http:/[*::*]80 Application started. Press Ctrl+C to shut down.”

Connect the global search service to Creatio

Actions on the server

To connect global search to Creatio, take the following steps on the server where the global search components are located (server 2 for the service deployed in Docker):

1. Install the api-get install curl or yum install curl utility for HTTP queries.

```
apt-get install curl
```

2. Execute the HTTP request to register the site in global search. Specify the following:
   a. [*DATABASE_TYPE*]: Creatio database type (mssql, postgresql, or oracle).
   b. [*DATABASE_CONNECTION_STRING*]: Creatio database connection string
   c. [*SITE_NAME*]: Creatio site name, e. g., my-test-site.
   d. [*SERVER2_IP_ADDRESS*] (only for Docker): the IP address of the Linux server where the global search components are deployed.

      [*GS_WEB_API_URL*] (only for Kubernetes): the IP address of the Linux server where the global search components are deployed.

      Request for Docker

      ```
      curl -v -X POST -d '{"databaseType": "[DATABASE_TYPE]", "databaseConnectionString": "[DATAB
      ```

      Request for Kubernetes

      ```
      curl -v -X POST -d '{"databaseType": "[DATABASE_TYPE]", "databaseConnectionString": "[DATAB
      ```

      Example for Microsoft SQL.

      For Docker:

      curl -v -X POST -d '{"databaseType": "mssql", "databaseConnectionString": "Server=myserver\\mssql2016; Database=my-test-site; User Id=my-login; Password='my-

password'; Connection Timeout=10"}' -H "Content-Type: application/json" http://[SERVER2_IP_ADDRESS]:81/sites/my-test-site

For Kubernetes:

curl -v -X POST -d '{"databaseType": "mssql", "databaseConnectionString": "Server=myserver\\mssql2016; Database=my-test-site; User Id=my-login; Password='my-password'; Connection Timeout=10"}' -H "Content-Type: application/json" http:// [GS_WEB_API_URL] /sites/my-test-site

Example for PostgreSQL. server=[SERVER_IP];port=5432;database=[DB_NAME];user id=[USER_NAME];password=[PASSWORD];timeout=10;commandtimeout=400;maxpoolsize=1024

3. Execute the HTTP request to connect the search to the site. Specify the following:

   a. [*SITE_NAME*]: Creatio site name, e. g., my-test-site.

   b. [*TEMPLATE_NAME*]: the name of the search template used in ElasticSearch. View the available templates in the table below.

   c. [*SERVER2_IP_ADDRESS*] (only for Docker): the IP address of the Linux server where the global search components are deployed.

   [*GS_WEB_API_URL*] (only for Kubernetes): the IP address of the Linux server where the global search components are deployed.

Request for Docker

```
curl -v -X POST -d '{"templateName": "[TEMPLATE_NAME]"}' -H "Content-Type: application/jso
```

Request for Kubernetes

```
curl -v -X POST -d '{"templateName": "[TEMPLATE_NAME]"}' -H "Content-Type: application/jso
```

Example.

For Docker:

curl -v -X POST -d '{"templateName": "default.json"}' -H "Content-Type: application/json" http://[SERVER2_IP_ADDRESS]:81/sites/my-test-site/search

For Kubernetes:

curl -v -X POST -d '{"templateName": "default.json"}' -H "Content-Type: application/json" http:// [GS_WEB_API_URL] /sites/my-test-site/search

Note. This request returns the URL of the index created in ElasticSearch. Save the URL and use it in the system setting installation SQL script below.

Attention. To change the search template, run the DELETE query at /sites/{siteName}/search, as well as the search connection HTTP request described above. This reindexes the entire site.

View the available search templates and their features in the table below:

	Old template (version 1.6)	default.json	ngram_2.json	ngram_3.json	without_ng
Search by partial match	+	–	+	+	–
Search by misspelled words	+	–	+	+	–
Search communication options by phone number (partial match)	+	+	+	+	–
Search communication options (partial match)	+	+	+	+	–
Search by word swapping	+	+	+	+	+
Search by exact match	+	+	+	+	+
Search by two characters	–	–	+	–	–
Average search speed (lower is better)		1x	13x	7x	<1x
Index size at elasticsearch (lower is better)		1x	4x	2.5x	<1x
Primary indexing time (lower is better)		1x	1.8x	1.4x	<1x

Settings in Creatio for Microsoft SQL DBMS

1. Toggle the global search (GlobalSearch, GlobalSearch_V2, GlobalSearchRelatedEntityIndexing) feature in Creatio by running the following SQL script:

```
DECLARE @GS_REIndexingFeature NVARCHAR(50) = 'GlobalSearchRelatedEntityIndexing';
DECLARE @GS_REIndexingFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE
Code = @GS_REIndexingFeature);

DECLARE @GlobalSearchFeature NVARCHAR(50) = 'GlobalSearch';
DECLARE @GlobalSearchFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code =

DECLARE @GlobalSearchV2Feature NVARCHAR(50) = 'GlobalSearch_V2';
DECLARE @GlobalSearchV2FeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code
DECLARE @allEmployeesId UNIQUEIDENTIFIER = 'A29A3BA5-4B0D-DE11-9A51-005056C00008';

IF (@GlobalSearchFeatureId IS NOT NULL)
 BEGIN
   IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @GlobalSearchFeatureId)
     UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId = @GlobalSearchFeature
    ELSE
     INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@all
 END;
ELSE
 BEGIN
   SET @GlobalSearchFeatureId = NEWID()
   INSERT INTO Feature (Id, Name, Code) VALUES (@GlobalSearchFeatureId, @GlobalSearchFeature,
   INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
 END;

IF (@GlobalSearchV2FeatureId IS NOT NULL)
 BEGIN
   IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @GlobalSearchV2FeatureId)
     UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId = @GlobalSearchV2Featu
    ELSE
     INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@all
 END;
ELSE
 BEGIN
   SET @GlobalSearchV2FeatureId = NEWID()
   INSERT INTO Feature (Id, Name, Code) VALUES (@GlobalSearchV2FeatureId, @GlobalSearchV2Feat
   INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
 END;

IF (@GS_REIndexingFeatureId IS NOT NULL)
BEGIN
 IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @GS_REIndexingFeatureId)
  UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId = @GS_REIndexingFeatureId
```

```
  ELSE
  INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEmp
END;
ELSE
BEGIN
 SET @GS_REIndexingFeatureId = NEWID()
 INSERT INTO Feature (Id, Name, Code) VALUES (@GS_REIndexingFeatureId, @GS_REIndexingFeature,
 INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEmpl
END;
```

2. Set the values of the system settings:

   a. “GlobalSearchUrl:” the full path to ElasticSearch, including the index. The web-api returns this value if you request it to add site search.

      Example string for Docker: http://[SERVER2_IP_ADDRESS]:83/indexname.

      Example string for Kubernetes: http://[GS-SEARCH-SERVICE_URL] /indexname.

   b. “GlobalSearchConfigServiceURL:” the global search API URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:81.

      Default value for Kubernetes: http:// [GS_WEB_API_URL].

   c. “GlobalSearchIndexingApiUrl:” the instant indexing service URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:82.

      Default value for Kubernetes: http://[GS-WEB-INDEXING-SERVICE_URL].

      For Docker

```
UPDATE SysSettingsValue
SET TextValue = [specify the URL to the ElasticSearch index, use a string of the following
WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchUrl')

UPDATE SysSettingsValue
SET TextValue = [specify the URL to the Global Search Service, use a string of the followi
WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchConfigServ

UPDATE SysSettingsValue
SET TextValue = [specify the URL to the Global Serch Indexing Service, use a string of the
WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchIndexingAp
```

      For Kubernetes

```
UPDATE SysSettingsValue

SET TextValue = [specify the URL to the ElasticSearch index, use a string of the following
```

```
WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchUrl')



UPDATE SysSettingsValue

SET TextValue = [specify the URL to the Global Search Service, use a string of the followi

WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchConfigSer



UPDATE SysSettingsValue

SET TextValue = [specify the URL to the Global Search Indexing Service, use a string of the

WHERE SysSettingsId = (SELECT TOP 1 Id FROM SysSettings WHERE Code ='GlobalSearchIndexingA
```

3. Restart Creatio, flush Redis, and log in to the application.

Settings in Creatio for Oracle DBMS

1. Toggle the global search (GlobalSearch, GlobalSearch_V2, GlobalSearchRelatedEntityIndexing) feature in Creatio by running the following SQL script:

```
CREATE OR REPLACE FUNCTION
generate_uuid return varchar2 is
    v_uuid varchar2(38);
    v_guid varchar2(32);
BEGIN
    v_guid := sys_guid();
    v_uuid := lower(
                '{' ||
                 substr(v_guid, 1,8) || '-' ||
                 substr(v_guid, 9,4) || '-' ||
                 substr(v_guid, 13,4) || '-' ||
                 substr(v_guid, 17,4) || '-' ||
                 substr(v_guid, 21) ||
                 '}'
                 );
    RETURN v_uuid;
END;
/

DECLARE
```

```
 GS_REIndexingFeature VARCHAR(50) := 'GlobalSearchRelatedEntityIndexing';
 GS_REIndexingFeatureId VARCHAR(38) := NULL;
 GS_REIndexingFeatureId_GUID VARCHAR(38) := generate_uuid();

GlobalSearchFeature VARCHAR(50) := 'GlobalSearch';
  GlobalSearchFeatureId VARCHAR(38) := NULL;
  GlobalSearchFeatureId_GUID VARCHAR(38) := generate_uuid();
  GlobalSearchV2Feature VARCHAR(50) := 'GlobalSearch_V2';
  GlobalSearchV2FeatureId VARCHAR(38) := NULL;
  GlobalSearchV2FeatureId_GUID VARCHAR(38) := generate_uuid();
  allEmployeesId VARCHAR(38) := '{7F3B869F-34F3-4F20-AB4D-7480A5FDF647}';
  State_GlobalSearch VARCHAR(1) := NULL;
  State_GlobalSearchV2 VARCHAR(1) := NULL;
  State_GS_REI VARCHAR(1) := NULL;

BEGIN
  SELECT MAX("Id") INTO GlobalSearchFeatureId FROM "Feature" WHERE "Code" = GlobalSearchFeatu
  SELECT MAX("Id") INTO GlobalSearchV2FeatureId FROM "Feature" WHERE "Code" = GlobalSearchV2F
SELECT MAX("Id") INTO GS_REIndexingFeatureId FROM "Feature" WHERE "Code" = GS_REIndexingFeatu

  SELECT MAX("FeatureState") INTO State_GlobalSearch FROM "AdminUnitFeatureState" WHERE "Feat
  SELECT MAX("FeatureState") INTO State_GlobalSearchV2 FROM "AdminUnitFeatureState" WHERE "Fe
  SELECT MAX("FeatureState") INTO State_GS_REI FROM "AdminUnitFeatureState" WHERE FeatureId"

  IF (GlobalSearchFeatureId IS NULL) THEN
      INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GlobalSearchFeatureId_GUID, Global
      INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VAL
    ELSE
      IF (State_GlobalSearch IS NOT NULL) THEN
          UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = GlobalSea
        ELSE
         INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId")
      END IF;
  END IF;

  IF (GlobalSearchV2FeatureId IS NULL) THEN
      INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GlobalSearchV2FeatureId_GUID, Glob
      INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VAL
    ELSE
      IF (State_GlobalSearchV2 IS NOT NULL) THEN
          UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = GlobalSea
        ELSE
          INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId")
      END IF;
  END IF;

 IF (GS_REIndexingFeatureId IS NULL) THEN
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GS_REIndexingFeatureId_GUID,GS_REIndex
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState","FeatureId") VALUES (
```

```
  ELSE
  IF (State_GS_REI IS NOT NULL) THEN
  UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" =GS_REIndexingFeatu
  ELSE
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState","FeatureId") VALUES (
  END IF;
 END IF;

END;
```

2. Set the values of the system settings:

   To do this, run the following script:

   a. “GlobalSearchUrl:” the full path to elasticsearch, including the index. The web-api returns this value if you request it to add site search.

      Example string for Docker: http://[SERVER2_IP_ADDRESS]:83/indexname.

      Example string for Kubernetes: http://[GS-SEARCH-SERVICE_URL] /indexname

   b. “GlobalSearchConfigServiceURL:” the global search API URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:81.

      Default value for Kubernetes: http:// [GS_WEB_API_URL].

   c. “GlobalSearchIndexingApiUrl:” the instant indexing service URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:82.

      Default value for Kubernetes: http://[GS-WEB-INDEXING-SERVICE_URL].

```
For Docker

DECLARE
  URL_SETTING_ID VARCHAR(38) := NULL;
  CONFIG_URL_SETTING_ID VARCHAR(38) := NULL;
  IND_API_SETTING_ID VARCHAR(38) := NULL;

  URL_VAL_ID VARCHAR(38) := NULL;
  CONFIG_URL_VAL_ID VARCHAR(38) := NULL;
  IND_API_VAL_ID VARCHAR(38) := NULL;

  SYS_ADMIN_UID VARCHAR(38) := '{A29A3BA5-4B0D-DE11-9A51-005056C00008}';

  ES_IND VARCHAR(500) := '[specify the URL to the ElasticSearch index, use a string of the fo
  CONFIG_URL VARCHAR(500) := '[specify the URL to the Global Search Service, use a string of
  IND_API_URL VARCHAR(500) := '[specify the URL to the Global Search Indexing Service, use a
BEGIN
  SELECT "Id" INTO URL_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchUrl';
  SELECT "Id" INTO CONFIG_URL_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchConfi
  SELECT "Id" INTO IND_API_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchIndexing
```

```
  SELECT MAX("Id") INTO URL_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = URL_SETTI
  SELECT MAX("Id") INTO CONFIG_URL_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = CO
  SELECT MAX("Id") INTO IND_API_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = IND_A

  IF (URL_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
        VALUES
        (URL_SETTING_ID, SYS_ADMIN_UID, '1', ES_IND);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = ES_IND WHERE "SysSettingsId" = URL_SETTING_
  END IF;

  IF (CONFIG_URL_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
        VALUES
        (CONFIG_URL_SETTING_ID, SYS_ADMIN_UID, '1', CONFIG_URL);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = CONFIG_URL WHERE "SysSettingsId" = CONFIG_U
  END IF;

  IF (IND_API_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
        VALUES
        (IND_API_SETTING_ID, SYS_ADMIN_UID, '1', IND_API_URL);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = IND_API_URL WHERE "SysSettingsId" = IND_API
  END IF;
END;
```

For Kubernetes

```
DECLARE

 URL_SETTING_ID VARCHAR(38) := NULL;
 CONFIG_URL_SETTING_ID VARCHAR(38) := NULL;
 IND_API_SETTING_ID VARCHAR(38) := NULL;

 URL_VAL_ID VARCHAR(38) := NULL;
 CONFIG_URL_VAL_ID VARCHAR(38) := NULL;
 IND_API_VAL_ID VARCHAR(38) := NULL;
```

```
 SYS_ADMIN_UID VARCHAR(38) := '{A29A3BA5-4B0D-DE11-9A51-005056C00008}';

 ES_IND VARCHAR(500) := '[specify the URL to the ElasticSearch index, use a string of the fol
 CONFIG_URL VARCHAR(500) := '[specify the URL to the Global Search Service, use a string of t
 IND_API_URL VARCHAR(500) := '[specify the URL to the Global Search Indexing Service, use a s

BEGIN
  SELECT "Id" INTO URL_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchUrl';
  SELECT "Id" INTO CONFIG_URL_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchConfi
  SELECT "Id" INTO IND_API_SETTING_ID FROM "SysSettings" WHERE "Code" = 'GlobalSearchIndexing

  SELECT MAX("Id") INTO URL_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = URL_SETTI
  SELECT MAX("Id") INTO CONFIG_URL_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = CO
  SELECT MAX("Id") INTO IND_API_VAL_ID  FROM "SysSettingsValue" WHERE "SysSettingsId" = IND_A

  IF (URL_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
        VALUES
        (URL_SETTING_ID, SYS_ADMIN_UID, '1', ES_IND);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = ES_IND WHERE "SysSettingsId" = URL_SETTING_
  END IF;

  IF (CONFIG_URL_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
        VALUES
        (CONFIG_URL_SETTING_ID, SYS_ADMIN_UID, '1', CONFIG_URL);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = CONFIG_URL WHERE "SysSettingsId" = CONFIG_U
 END IF;
 IF (IND_API_VAL_ID IS NULL)
    THEN
      INSERT INTO "SysSettingsValue"
        ("SysSettingsId", "SysAdminUnitId", "IsDef", "TextValue")
      VALUES
       (IND_API_SETTING_ID, SYS_ADMIN_UID, '1', IND_API_URL);
    ELSE
      UPDATE "SysSettingsValue" SET "TextValue" = IND_API_URL WHERE "SysSettingsId" = IND_API
  END IF;
END;
```

3. Restart Creatio, flush Redis, and log in to the application.

Settings in Creatio for PostgreSQL DBMS

1. Toggle the global search (GlobalSearch, GlobalSearch_V2, GlobalSearchRelatedEntityIndexing) feature in Creatio by running the following SQL script:

```
DO $$

DECLARE
    GlobalSearchFeature VARCHAR(50) := 'GlobalSearch';
    GlobalSearchFeatureId uuid;
    GlobalSearchV2Feature VARCHAR(50) := 'GlobalSearch_V2';
    GlobalSearchV2FeatureId uuid;
    GS_RelatedEntityIndexingFeature VARCHAR(50) :=   'GlobalSearchRelatedEntityIndexing';
    GS_RelatedEntityIndexingFeatureId uuid;
    allEmployeesId uuid := 'A29A3BA5-4B0D-DE11-9A51-005056C00008';

BEGIN

   SELECT "Id" INTO GlobalSearchFeatureId FROM "Feature"
   WHERE "Code" = GlobalSearchFeature
   LIMIT 1;
   IF (GlobalSearchFeatureId IS NOT NULL)
      THEN
          IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = GlobalSearchFe
             UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = Global
          ELSE
              INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "Feature
          END IF;
   ELSE
       GlobalSearchFeatureId := uuid_generate_v4();
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GlobalSearchFeatureId, GlobalSear
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VA
   END IF;

   SELECT "Id" INTO GlobalSearchV2FeatureId FROM "Feature"
   WHERE "Code" = GlobalSearchV2Feature
   LIMIT 1;
   IF (GlobalSearchV2FeatureId IS NOT NULL)
    THEN
        IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = GlobalSearchV2Fe
            UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = GlobalS
        ELSE
           INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId"
        END IF;
   ELSE
       GlobalSearchV2FeatureId := uuid_generate_v4();
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GlobalSearchV2FeatureId, GlobalSe
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VA
```

```
    END IF;

  SELECT "Id" INTO GS_RelatedEntityIndexingFeatureId FROM "Feature" WHERE "Code" =GS_RelatedE
  IF (GS_RelatedEntityIndexingFeatureId IS NOT NULL)
 THEN
    IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = GS_RelatedEntityInde
UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = GS_RelatedEntityInd
  ELSE
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState","FeatureId") VALUES (
  END IF;
  ELSE
  GS_RelatedEntityIndexingFeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (GS_RelatedEntityIndexingFeatureId, GS_
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState","FeatureId") VALUES (
  END IF;
END $$;
```

2. Set the values of the system settings:

   a. “GlobalSearchUrl:” the full path to ElasticSearch, including the index. The web-api returns this value if you request it to add site search.

      Example string for Docker: http://[SERVER2_IP_ADDRESS]:83/indexname.

      Example string for Kubernetes: http://[GS-SEARCH-SERVICE_URL] /indexname.

   b. “GlobalSearchConfigServiceURL:” the global search API URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:81.

      Default value for Kubernetes: http:// [GS_WEB_API_URL].

   c. “GlobalSearchIndexingApiUrl:” the instant indexing service URL.

      Default value for Docker: http://[SERVER2_IP_ADDRESS]:82.

      Default value for Kubernetes: http://[GS-WEB-INDEXING-SERVICE_URL].

   To do this, run the following script:

```
For Docker

UPDATE "SysSettingsValue"
SET "TextValue" = [specify the URL to the ElasticSearch index, use a string of the following
WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchUrl' LIMI

UPDATE "SysSettingsValue"
SET "TextValue" = [specify the URL to the Global Search Service, use a string of the followin
WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchConfigSer

UPDATE "SysSettingsValue"
SET "TextValue" = [specify the URL to the Global Search Indexing Service, use a string of the
WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchIndexingA
```

```
For Kubernetes

UPDATE "SysSettingsValue"

SET "TextValue" = [specify the URL to the ElasticSearch index, use a string of the following

WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchUrl' LIMI



UPDATE "SysSettingsValue"

SET "TextValue" = [specify the URL to the Global Search Service, use a string of the followin

WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchConfigSer



UPDATE "SysSettingsValue"

SET "TextValue" = [specify the URL to the Global Search Indexing Service, use a string of the

WHERE "SysSettingsId" = (SELECT "Id" FROM "SysSettings" WHERE "Code" = 'GlobalSearchIndexingA
```

3. Restart Creatio, flush Redis, and log in to the application.

Global search and deduplication FAQ

PRODUCTS: ALL CREATIO PRODUCTS

How do I check which version of the global search service I have?

If you use Creatio cloud, you will always have the latest version of the global search service.

If you use Creatio on-site, run the following console command:

```
docker ps
```

This will open the list of all running containers. The number of global search version will be available in the [*image*] column.

How do I initiate the re-indexing of my Creatio site?

For global search version 2.0 and later:

Execute the PUT request in this format:

```
http://[ GS-WEB-API ]:81/indexation/{siteName}/reindex/full
```

For global search versions earlier than 2.0:

1. Open gs-mysql container via the following command:

   ```
   docker exec -it gs-mysql bash
   ```

2. Run the following command in the opened gs-mysql container:

   ```
   mysql -p1665017 use gs; UPDATE GlobalSearchIndexingEntity SET LastIndexedOn = NULL, InProcess
   ```

How do I enable the global search service logging?

By default, the service logs only errors. To enable logging of all events, locate the following string in the docker-compose.yaml file:

```
-Log4NetPath=${LOG$NET_CONFIG_FILE:-log4net.production.config}
```

Replace the string with the following string:

```
-Log4NetPath=${LOG$NET_CONFIG_FILE:-log4net.debug.config}
```

Note. If you enable logging of all events, the number of log files will increase significantly.

Which metrics and tracking schema can I use to monitor the global search operation?

Execute the following request:

```
http://[ GS-WEB-API ]:81/sites/[ SITE_NAME ]/search/state
```

Here, [*GS-WEB-API*] is the server address, and [*SITE_NAME*] is your Creatio website name.

How do I set up access to ElasticSearch via a password?

You can restrict access to ElasticSearch using Haproxy that supports base64 authentification. Use the x-pack plugin to set up access to ElasticSearch via a login and a password.

How do I add a new object to the ElasticSearch indexing, or change the settings for the indexed fields of existing objects?

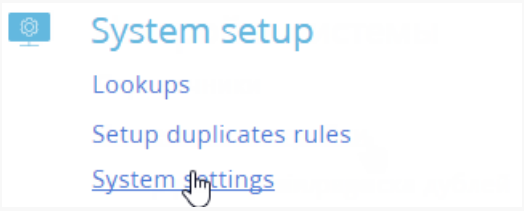
In **Creatio version 7.18.4 and earlier**, you can enable and configure indexing of specific sections using Creatio in-app tools. By default, ElasticSearch will index only sections regardless of their author, as well as string and lookup columns (with a few exceptions). View the up-to-date list of exceptions in the attached *.pdf file.

Since **Creatio version 7.18.5**, you can optimize columns indexed for global search. To improve Creatio performance and reduce the load on the server, you can specify the columns to exclude from global search indexing explicitly. For example, columns that contain service information.

To modify the list of indexed columns:

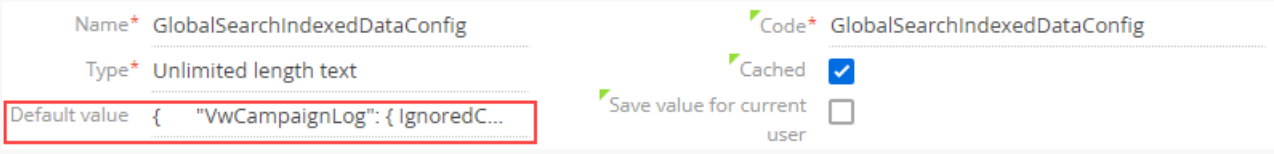
1. Open the System Designer. For example, click the button.
2. Click [*System settings*] in the [*System setup*] block (Fig. 1).

Fig. 1 The [*System settings*] section



3. Select the “GlobalSearchIndexedDataConfig” (“GlobalSearchIndexedDataConfig” code) system setting in the list and click the [*Open*] button.
4. The [*Default value*] field on the system setting page contains the JSON list of columns to exclude from indexing (Fig. 2).

Fig. 2 The field that contains the list of columns to exclude from indexing



5. Copy the list of columns to exclude from indexing and edit it externally. You can do it in any online JSON editor.

   The list of columns to exclude from indexing consists of blocks. Each block contains a unique object code, which you can find in the [*Code*] field in the [*Configuration*] section. View an example of the block structure below:

```
"Contact":{ // the object code

"IgnoredColumns":[ // the list of columns to exclude from indexing

"dear", // the code of the column to exclude from indexing

"salutationtype",

"gender",...]   },
```

Note. If you specify “*” instead of the column (object) name, Creatio excludes all columns (objects) from indexing. If you add underscore to the column name, Creatio excludes all columns whose names end in the fragment marked by the underscore from indexing. For example, the “_headerproperties” column name excludes all columns that end in “headerproperties” from indexing.

6. Validate the JSON string you edited, for example, in an online editor.
7. Paste the updated list to the [*Default value*] field and save the changes.
8. Restart the indexing to force update the column list in ElasticSearch.

How do I deploy ElasticSearch on several servers with a single URL? How do I set up clustering?

Learn more in the Elastic service documentation.

Why is the “Duplicates search rules” setting not displayed for me?

Check if the “Deduplication service api address” (“DeduplicationWebApiUrl” code) system setting is populated and whether the following features are enabled in “FeatureToggle:”

- “BulkESDeduplication”
- “ESDeduplication”
- “Deduplication”

Learn more about enabling additional functionality in developer documentation: Feature Toggle mechanism.

Can I use the global search and bulk duplicate search services in two Creatio applications simultaneously?

If you are using two Creatio applications, for example, production and developer environments, you can set up the global search and bulk duplicate search services for each of them independently. Use the following guides to set up the services:

- Global search.
- Bulk duplicate search.

How does Creatio sort the search results?

The display order of the search results mainly depends on their relevance. The following factors affect relevance:

- The volume of text in the document.
- The frequency of the search query in the document.
- The frequency of the search query in the index, as well as other, less important, parameters.

The following system settings affect the display order of search results as well:

- “Global search default entity weight” (“GlobalSearchDefaultEntityWeight” code). Increases the display priority of section records where the search was performed. For example, if you enter a search query from the [*Contacts*] section, the records of this section will appear first in the list.
- “Global search default primary column weight” (“GlobalSearchDefaultPrimaryColumnWeight” code). Increases the display priority of specific search results. It applies to records whose primary column value matches the search query. For example, [*Full name*] is a primary column for the contact, and [*Name*] is a primary column for the account. If the search query matches the value in the record’s primary column, this record will be displayed at the top of the search results.

While these system settings affect relevance, they do not guarantee the exact display order of search results as the other factors affect the order as well.

Do microservices support Windows authentication?

Since you must deploy the global search and deduplication service using Docker, microservices do not support Windows authentication.

Bulk duplicate search

PRODUCTS: ALL CREATIO PRODUCTS

Use bulk duplicate search to deduplicate Creatio section records.

Attention. Set up the global search service in ElasticSearch to ensure correct operation of the bulk duplicate search. Learn more in a separate article: Global search. You need repository access to set up the current version of the global search service. Contact Creatio support to verify your license and gain access to the repository.

Basic knowledge of kubernetes or docker-compose and Linux administration is required to set up bulk duplicate search service.

You can deploy the components of global duplicate search using **Kubernetes orchestrator and Helm package manager** or **Docker**.

Use the requirements calculator to check the server requirements.

Set up the bulk duplicate search service using Kubernetes

To set up the service, download the source files. Download files.

To install the service:

1. Set up the target environment:
   a. **Kubernetes cluster**. Learn more about setting up and managing the cluster in the Kubernetes documentation.
   b. **Helm package manager**. Learn more about installing the package manager in the Helm documentation.
   c. **Global search service** via Kubernetes. Learn more in a separate article: Global search.
2. Unpack the **values-onsite.yaml** file from the source file archive and save the file to the same directory as the archive.
3. Open the file. View the main parameters the file uses in the table below. To optimize the load on Redis and RabbitMQ, we recommend using the same service instances for both global search and bulk duplicate search. Bulk duplicate search always uses the same ElasticSearch service as global search.

   Note. By default, the Mongodb service is deployed together with the duplicate search service. If you need to change the Mongodb deployment parameters, unpack the source files from the archive and change the parameters in the mongodb section of the values-onsite.yaml file.

4. Set up the connection to services you installed previously, for example, when setting up global search, in the **values-onsite.yaml** source file.

   - For **Redis**:

     Set up the connection to the previously installed Redis. To do this, specify the connection parameters in the `global.redis` section of the **values-onsite.yaml** file.

     ```
     # Multi pods global parameters
     global:
       # Redis server connection parameters
       redis:
         host: [host]
         port: [port]
         database: [database]
     ```

     Where

     [host] is the address of the Redis server.

     [port] is the connection port of the Redis server.

     [database] is the name of the Redis database.

   - For **RabbitMQ**:

Set up the connection to the previously installed RabbitMQ. To do this, specify the connection parameters in the `global.rabbitmq` section of the **values-onsite.yaml** file.

```
# Multi pods global parameters
global:
  # RabbitMQ connection parameters
  rabbitmq:
    host: [host]
    vhost: [vhost]
    port: [port]
    user: [user]
    password: [password]
```

Where

[host] is the address of the RabbitMQ service.

[vhost] is the virtual host address of the RabbitMQ service.

[port] is the amqp connection port of RabbitMQ.

[user] is the RabbitMQ user.

[password] is the RabbitMQ user password.

- For **Mongodb**:
  - Disable the Mongodb installation. To do this, specify `enabled: false` in the `mongodb` section of the **values-onsite.yaml** file.

    ```
    mongodb:
      enabled: false
    ```

  - Set up the connection to external mongodb. To do this, specify the connection parameters in the `global.mongodb` section of the **values-onsite.yaml** file.

    ```
    # Multi pods global parameters
    global:
      # Deduplication database connection parameters
      mongodb:
        host: [host]
        port: [port]
        user: [user]
        password: [password]
    ```

    Where

    [host] is the address of the Mongodb service.

    [port] is the port to connect to the service.

[user] is the Mongodb user on whose behalf to connect.

[password] is the Mongodb user password.

- For **ElasticSearch**:

  Set up the connection to the previously installed ElasticSearch. To do this, specify the connection parameters in the `global.elasticsearch` section of the **values-onsite.yaml** file.

```
# Multi pods global parameters
global:
  # Elastic search connection parameters
  elasticsearch:
    protocol: [protocol]
    host: [host]
    port: [port]
    path: [path]
    user: [user]
    password: [password]
```

  Where

  [user] is the ElasticSearch user.

  [password] is the ElasticSearch user password.

  [port] is the port to connect to ElasticSearch.

  [path] is the path parameter of the ElasticSearch service (by default, \).

  [protocol] is the ElasticSearch connection protocol (by default, http).

  [host] is the address of the ElasticSearch service.

5. Run the `helm install gs -f values-onsite.yaml deduplication.tgz` command. As a result, Helm will install the bulk duplicate search service and selected dependencies.

> **Note.** By default, Helm deploys services with NodePort type.

The main parameters of the global search service the **values.yaml** file uses.

Parameter	Parameter description
duplicatesSearchWorker.maxDuplicatesPerRecord	The maximum allowable number of duplicates for a single record.
log4Net	Logging settings.
global.elasticsearch	ElasticSearch connection parameters.
global.rabbitmq	RabbitMQ connection settings.
global.mongodb	The connection settings of the duplicate search service's internal base.
global.db	The connection settings of the global search service's internal service database.
global.redis	Redis connection settings.

Set up the bulk duplicate search service in Docker

Components of the bulk duplicate search service

Prerequisites:

1. Global search components. View the index in a separate article: Global search.
2. Components of the bulk duplicate search service. View the component index below.

Mongodb. Document-oriented DBMS.

dd-web-api. Web service for communication in Creatio.

dd-data-service. Internal service for communication with mongodb.

dd-duplicates-search-worker. Duplicate search component.

dd-duplicates-deletion-worker. Component for targeted deletion of duplicates.

dd-duplicates-confirmation-worker. Component that groups and filters found duplicates, taking into account their uniqueness.

dd-duplicates-cleaner. Component that clears duplicates.

dd-deduplication-task-worker. Component that sets the deduplication task.

dd-deduplication-preparation-worker. Component that prepares the deduplication process. Generates duplicate search queries according to the rules.

dd-deduplication-task-diagnostic-worker. Component that controls the execution of the duplicate search task.

To set up the components, download the source files. Download files.

1. Deploy and set up Creatio global search.

2. Download and unpack the necessary source files. Copy them to the computer that has docker, docker-compose software installed. Download files.
3. Set up the environment variables.
4. Launch the containers.
5. Verify that containers are running successfully.
6. Verify logging.
7. Enable the bulk duplicate search functionality in Creatio.

Set up the environment variables

The compose/.env file stores the environment variables. Edit this file to set the variables.

Variable name	Details	Default value
ELASTICSEARCH_URI	The IP address of the server where you deployed ElasticSearch as part of Creatio global search setup.	http://user:password@external.elasticsearch-ip:9200/

Launch the containers

To launch the containers, run the following command:

```
cd compose # go to the compose directory
docker-compose up -d
```

Verify that containers ran successfully

To view the list of all running containers, run the following command at the console:

```
docker ps --filter "label=service=dd" -a --format "table {{.Names}}\t{{.Ports}}\t{{.Status}}\t{{
```

The running containers have an “Up” status.

Verify logging

By default, logging is performed during the “stdout” container command execution. To view the last 100 records from the dd-data-service container, run the following command:

```
docker logs --tail 100 dd-data-service
```

Update the bulk duplicate search version

To update the bulk duplicate search version 2.0 to version 3.0, take the following steps.

1. Back up Creatio duplicate data. To do this, run the `/api/snapshot/backup/gzip/{indexName}` command via the http://[server IP address]:8086/api swagger web-api. {indexName} is the name (last 64 characters) of the index in the “Global search url address” (“GlobalSearchUrl” code) system setting.
2. Delete the version 2.0 of docker volumes. To do this, open the docker-compose directory that contains the version 2.0 files and run the docker-compose down -v command.
3. Install the version 3.0 of the bulk duplicate search service.
4. Upload the duplicate data retrieved on step 1 to the service. To do this, run the `/api/snapshot/restore/gzip` command in the new service version via swagger.

Enable the bulk duplicate search functionality in Creatio

Take the following steps in Creatio.

1. Configure the “Deduplication service api address” system setting.
2. Set up the “Duplicates search” operation permissions.
3. Enable the bulk duplicate search functionality in Creatio. Note that this setting is DBMS-specific.
4. Restart the Creatio application.

Configure the “Deduplication service api address” system setting

Go to the [*System settings*] section, open the “Deduplication service api address” (“DeduplicationWebApiUrl” code) system setting, and specify the URL to dd-web-api. Use a string of the following type: http://external.deduplication-web-api:8086.

Set up the “Duplicates search” operation permissions

Go to the [*Operation permissions*] section, open the “Duplicates search” (“CanSearchDuplicates” code) system operation, and, on the [*Operation permission*] detail, grant permissions to the necessary users/roles, who can search for duplicates.

Enable the bulk duplicate search functionality in Creatio

Toggle the bulk duplicate search (Deduplication, ESDeduplication, BulkESDeduplication) functionality by running an SQL script. The script depends on the DBMS: Microsoft SQL, Oracle, or PostgreSQL.

```
For Microsoft SQL DBMS

DECLARE @DeduplicationFeature NVARCHAR(50) = 'Deduplication';
DECLARE @DeduplicationFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code = @D

DECLARE @ESDeduplicationFeature NVARCHAR(50) = 'ESDeduplication';
DECLARE @ESDeduplicationFeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id FROM Feature WHERE Code =
```

```
DECLARE @Bulk_ES_DD_Feature NVARCHAR(50) = 'BulkESDeduplication';
DECLARE @Bulk_ES_DD_FeatureId UNIQUEIDENTIFIER = (SELECT TOP 1 Id
FROM Feature WHERE Code =@Bulk_ES_DD_Feature);

DECLARE @allEmployeesId UNIQUEIDENTIFIER = 'A29A3BA5-4B0D-DE11-9A51-005056C00008';
IF (@DeduplicationFeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @DeduplicationFeatureId)
       UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId =@DeduplicationFeatureI
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
@DeduplicationFeatureId)
END;
ELSE
BEGIN
      SET @DeduplicationFeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES
(@DeduplicationFeatureId, @DeduplicationFeature, @DeduplicationFeature)
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
END;

IF (@ESDeduplicationFeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @ESDeduplicationFeatureId
     UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId = @ESDeduplicationFeature
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allE
END;
ELSE
BEGIN
      SET @ESDeduplicationFeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES (@ESDeduplicationFeatureId, @ESDeduplicationFe
      INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
END;

IF (@Bulk_ES_DD_FeatureId IS NOT NULL)
BEGIN
      IF EXISTS (SELECT * FROM AdminUnitFeatureState WHERE FeatureId = @Bulk_ES_DD_FeatureId)
       UPDATE AdminUnitFeatureState SET FeatureState = 1 WHERE FeatureId =@Bulk_ES_DD_FeatureId
      ELSE
       INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState,FeatureId) VALUES (@allEm
END;
ELSE
BEGIN
      SET @Bulk_ES_DD_FeatureId = NEWID()
      INSERT INTO Feature (Id, Name, Code) VALUES (@Bulk_ES_DD_FeatureId, @Bulk_ES_DD_Feature, @
      INSERT INTO AdminUnitFeatureState (SysAdminUnitId, FeatureState, FeatureId) VALUES (@allEm
END;
```

For Oracle DBMS

```
CREATE OR REPLACE FUNCTION
generate_uuid return varchar2 is
       v_uuid varchar2(38);
       v_guid varchar2(32);
BEGIN
       v_guid := sys_guid();
       v_uuid := lower(
'{' ||
       substr(v_guid, 1,8) || '-' ||
       substr(v_guid, 9,4) || '-' ||
       substr(v_guid, 13,4) || '-' ||
       substr(v_guid, 17,4) || '-' ||
       substr(v_guid, 21) ||
       '}'
       );
       RETURN v_uuid;
END;
/
DECLARE
      DeduplicationFeature VARCHAR(50) := 'Deduplication';
      DeduplicationFeatureId VARCHAR(38) := NULL;
      DeduplicationFeatureId_GUID VARCHAR(38) := generate_uuid();
      ESDeduplicationFeature VARCHAR(50) := 'ESDeduplication';
      ESDeduplicationFeatureId VARCHAR(38) := NULL;
      ESDeduplicationFeatureId_GUID VARCHAR(38) := generate_uuid();
      BulkESDeduplicationFeature VARCHAR(50) := 'BulkESDeduplication';
      BulkESDeduplicationFeatureId VARCHAR(38) := NULL;
      Bulk_ES_DD_GUID VARCHAR(38) := generate_uuid();
      allEmployeesId VARCHAR(38) := '{7F3B869F-34F3-4F20-AB4D-7480A5FDF647}';
      State_Deduplication VARCHAR(1) := NULL;
      State_ESDeduplication VARCHAR(1) := NULL;
      State_BulkESDeduplication VARCHAR(1) := NULL;
BEGIN
      SELECT MAX("Id") INTO DeduplicationFeatureId FROM "Feature" WHERE "Code" = DeduplicationFe
      SELECT MAX("Id") INTO ESDeduplicationFeatureId FROM "Feature" WHERE "Code" = ESDeduplicati
      SELECT MAX("Id") INTO BulkESDeduplicationFeatureId FROM "Feature" WHERE "Code" = BulkESDed
      SELECT MAX("FeatureState") INTO State_Deduplication FROM "AdminUnitFeatureState" WHERE "Fe
      SELECT MAX("FeatureState") INTO State_ESDeduplication FROM "AdminUnitFeatureState" WHERE "
      SELECT MAX("FeatureState") INTO State_BulkESDeduplication FROM "AdminUnitFeatureState" WHE
      IF (DeduplicationFeatureId IS NULL) THEN
       INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (DeduplicationFeatureId_GUID, Dedupli
       INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
       ELSE
```

```
        IF (State_Deduplication IS NOT NULL) THEN
        UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = DeduplicationFe
        ELSE
        INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
        END IF;
END IF;
IF (ESDeduplicationFeatureId IS NULL) THEN
        INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (ESDeduplicationFeatureId_GUID, ESDed
        INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
        ELSE
        IF (State_ESDeduplication IS NOT NULL) THEN
        UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = ESDeduplicatior
        ELSE
        INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
        END IF;
END IF;
IF (BulkESDeduplicationFeatureId IS NULL) THEN
        INSERT INTO "Feature" ("Id", "Name", "Code") VALUES(Bulk_ES_DD_GUID, BulkESDeduplicationF
        INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
        ELSE
        IF (State_BulkESDeduplication IS NOT NULL) THEN
        UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = BulkESDeduplica

        ELSE
        INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUE
        END IF;
      END IF;
END;
```

For PostgreSQL DBMS

```
DO $$

DECLARE
 DeduplicationFeature VARCHAR(50) := 'Deduplication';
 DeduplicationFeatureId uuid;

 ESDeduplicationFeature VARCHAR(50) := 'ESDeduplication';
 ESDeduplicationFeatureId uuid;

 Bulk_ES_DD_Feature VARCHAR(50) := 'BulkESDeduplication';
 Bulk_ES_DD_FeatureId uuid;

  allEmployeesId uuid := 'A29A3BA5-4B0D-DE11-9A51-005056C00008';

BEGIN
```

```
 SELECT "Id" INTO DeduplicationFeatureId FROM "Feature"
 WHERE "Code" = DeduplicationFeature
 LIMIT 1;
 IF (DeduplicationFeatureId IS NOT NULL)
  THEN
   IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = DeduplicationFeatureId)
    UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = DeduplicationFeatu
   ELSE
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
   END IF;
 ELSE
  DeduplicationFeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (DeduplicationFeatureId, DeduplicationFeat
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
 END IF;

 SELECT "Id" INTO ESDeduplicationFeatureId FROM "Feature"
 WHERE "Code" = ESDeduplicationFeature
 LIMIT 1;
 IF (ESDeduplicationFeatureId IS NOT NULL)
  THEN
   IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = ESDeduplicationFeatureId
    UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = ESDeduplicationFea
   ELSE
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
   END IF;
 ELSE
  ESDeduplicationFeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (ESDeduplicationFeatureId, ESDeduplicatior
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
 END IF;

 SELECT "Id" INTO Bulk_ES_DD_FeatureId FROM "Feature"
 WHERE "Code" = Bulk_ES_DD_Feature
 LIMIT 1;
 IF (Bulk_ES_DD_FeatureId IS NOT NULL)
  THEN
   IF EXISTS (SELECT * FROM "AdminUnitFeatureState" WHERE "FeatureId" = Bulk_ES_DD_FeatureId) TH
    UPDATE "AdminUnitFeatureState" SET "FeatureState" = 1 WHERE "FeatureId" = Bulk_ES_DD_Feature
   ELSE
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (
   END IF;
 ELSE
  Bulk_ES_DD_FeatureId := uuid_generate_v4();
  INSERT INTO "Feature" ("Id", "Name", "Code") VALUES (Bulk_ES_DD_FeatureId, Bulk_ES_DD_Feature,
  INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId") VALUES (al
 END IF;
END $$;
```

Restart the Creatio application

Clear redis, restart the Creatio application and log in.

Recommended operations for the service functioning

We recommend performing mongodb backup once a day to support the functionality of the service and enable restoring of data, e. g., in case of electricity breakdowns.

Machine learning service

PRODUCTS: ALL CREATIO PRODUCTS

Machine learning service predicts values based on large volumes of historical data and current facts. Learn more in a separate article: Predictive data analysis.

Attention. Base knowledge of Docker, Linux or Windows administration is required to set up the machine learning service.

Getting started

Note. Creatio on-site requires the “creatio predictive service on-site” license for the service to operate as intended. If you need to purchase the license, contact the responsible manager.

To set up the service, you need to have a server (physical or virtual machine) with Linux or Windows OS installed. Docker software is used for installing the service components. Download the archive containing the configuration files and installation scripts. Download archive. Depending on your company needs, you can use either Docker Community Edition (CE) or Enterprise Edition (EE). Learn more in the Docker Guide.

Attention. We recommend using a Linux-based server for production environment. You can only use a Windows based server for the development environment. Contact the support service to receive Docker containers that care compatible with Windows.

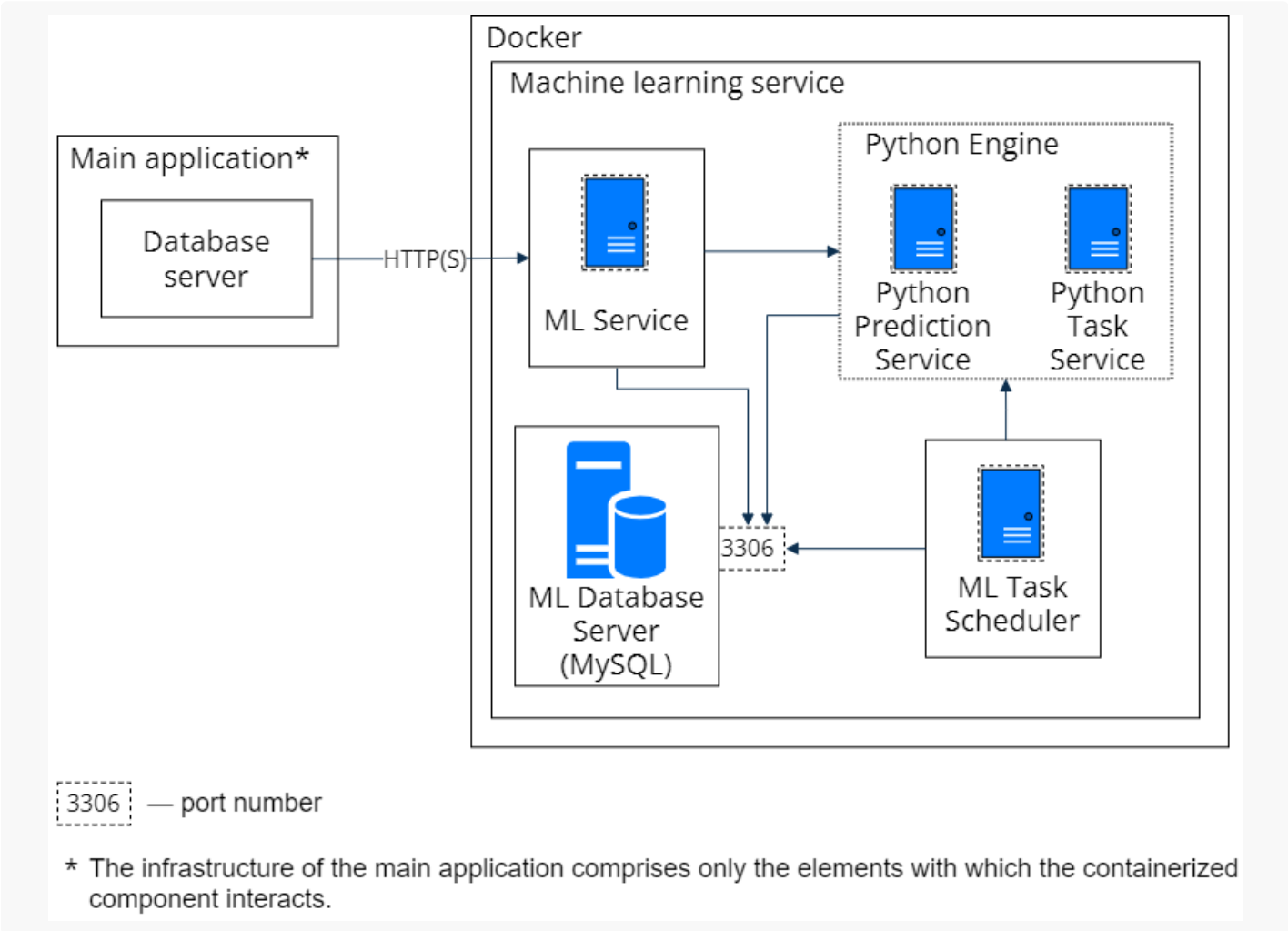
Use the requirements calculator to check the server requirements.

Machine learning service components

The machine learning service uses the following components (Fig. 1):

- **ML Service**. Machine learning web service. The only component enabling external access.
- **Python Engine**. Machine learning wrapper service for open-source machine learning libraries.
- **ML Task Scheduler**. Task scheduler.
- **MySQL**. MySQL database. You can access it via the standard 3306 port.

Fig. 1 Machine learning service components



All the components are packed as Docker images for the convenient on-site installation of the service.

Set up machine learning service

Follow this procedure to set up the machine learning service:

1. Install Docker. Read more >>>
2. Install Docker Compose. Read more >>>
3. Install and set up the service components. Read more >>>
4. Verify the installation. Read more >>>

Install Docker

Docker installation on Linux platforms is covered in the Docker guide.

Run the “docker --version” command on Linux machine to verify the installed Docker version.

Install Docker Compose

Docker Compose installation on Linux platforms is covered in the Docker guide.

Run the “docker-compose --version” command on Linux machine to verify the installed Docker Compose version.

Set up the machine learning service components

Deploy the containers of the machine learning service components via the Docker Compose utility. Download the configuration files and scripts that are necessary to deploy and configure the service components. Download the archive

> **Note.** The configuration files contain all necessary default settings for a Linux based server.

The **archive structure of the configuration files and scripts** is as follows:

/etc/

../ml-service/appsettings.json. The ML web service configuration.

../ml-service/log4net.config. The setup of the web service logging level.

../task-scheduler/appsettings.json. The “ML Task Scheduler” utility configuration.

../task-scheduler/log4net.config. The setup of “ML Task Scheduler” logging level.

Docker-compose.yml. The “Docker Compose” utility configuration.

.env. The file that contains environment variables for running the components. For example, MySQL password.

> **Attention.** If you need to change the password to MySQL database, update it in the .env file and other configuration files that contain database access setup sections.

Install the machine learning service components

1. Download and unzip the archive with the configuration files and scripts to a custom directory, for example, /opt/ml.
2. Using the Linux terminal, go to the /docker-compose catalog of the unzipped archive, for example, /opt/ml/docker-compose.
3. Run the “sudo docker-compose pull” command in the terminal. Wait until the download of the required service component images from the Docker Hub is complete.

> **Attention.** If the server is disconnected from the Internet, download all required images to an Internet-

connected computer manually (see the “docker-compose.yml” configuration file). Then use the sudo docker export and sudo docker import commands to transfer the images to the target computer as files.

4. Run the **sudo docker-compose run dbmigration** command to initialize the database structure. Wait until the command execution is complete.
5. Run the **sudo docker-compose up -d** command to launch the services. A “logs” folder will be created in the current catalog.

Verify the setup of the machine learning service components

1. To verify the installation of ML web service, run the following command in Linux:

   curl -X GET localhost:5005/readiness

   The service must return the following response:

   Healthy
2. To verify the running of ML Task Scheduler, execute the following command in the Linux terminal:

   curl -X GET localhost:5004/readiness

   The service must return the following response:

   Healthy
3. To verify the running of R Engine, execute the following command in the Linux terminal:

   curl -X GET localhost:8081/readiness

   The service must return the following response:

   R Service is ready
4. To verify creating of tables, run the following command in the terminal:

   docker exec -it DB Container Id mysql -u root --password=Supervisor ml -e "show tables;"

   where [*DB Container Id*] is an identifier of the container with a database component. You can find out the container identifier using the **sudo docker ps** command.

   Example. Verification of creating tables:

   docker exec -it [*DB Container Id*] mysql -u root --password=Supervisor ml -e "show tables;"

   As a result, the names of primary service tables should be displayed: “modelinstance”, “traindata”, “trainsession”, etc.

Perform the setup in Creatio

To work with the prediction service in Creatio, fill out the following settings:

1. “Creatio cloud services API key” (“CloudServicesAPIKey” code) system setting. Cloud services need it to authenticate your Creatio instance.
2. “Periodicity of machine learning model training job” (“MLModelTrainingPeriodMinutes” code) system setting. It determines the models' synchronization launch frequency.

3. Add the prediction service's URL to the [*Service endpoint Url*] field for all the records in the [*ML problem types*] lookup.

Update the machine learning service components

Attention. We recommend saving a backup copy of MySQL database, before you update the services. Learn more in the Docker Guide.

1. Using the Linux terminal, go to the docker-compose catalog with the configured files, for example, /opt/ml/docker-compose.
2. Run the **sudo docker-compose stop** command to stop the service component containers.
3. Run the **sudo docker-compose pull** command in the terminal. Wait until the download of the required service component images from the Docker Hub is complete.
4. Run the **sudo docker-compose run dbmigration** command to initialize the database structure. Wait until the command execution is complete.
5. Run the **sudo docker-compose up -d** command to launch the services.

Attention. If your application already has configured and trained data models, we recommend retraining them after updating the service.

Email Listener synchronization service

PRODUCTS: ALL CREATIO PRODUCTS

The Email Listener (formerly Exchange Listener) service synchronizes Creatio with Microsoft Exchange and IMAP/SMTP mail services using a subscription mechanism. Email Listener lets you use horizontal scaling that enables the active use of email synchronization block and controlled use of resources.

The synchronization service is required to manage emails in Creatio .NET Framework and .NET Core since version 7.17.2. This article covers the deployment of Exchange Listener synchronization service for Creatio on-site.

The service consists of the following components:

1. **Email Listener (EL API)**. Initiates an outgoing connection to EWS API or IMAP. Creates a subscription to “new message” events using the mailbox credentials. The open subscription remains in the component memory to ensure fast response time when new emails arrive. The email is downloaded upon receiving the corresponding event. An in-memory repository is sufficient to deploy the service. A required service component.
2. **NoSQL Redis DBMS**. Creates a scalable and fault tolerant system of handler nodes. The Redis repository holds information about the mailboxes that are served. This enables any container to handle Creatio requests to add a new subscription or check the status of a specific subscription regardless of the subscription node. Redis requires a separate database for the Exchange Listener service operation. A required service component.

3. **Email Worker (EL Worker)**. Maintains the scalability and fault tolerance of the primary Email Listener module. The additional module downloads emails from the mail server and delivers them to Creatio. This enables high-load services to handle emails smoother during peaks in the email flow. The EL Worker reduces the load on the EL API components that no longer need to download emails. Instead, the components can manage the subscription and send outgoing emails.
4. **RabbitMQ**. Maintains the scalability and fault tolerance of the service. The queue broker distributes tasks between components in high-load environments.

Determine the configuration of the Email Listener service

Determine the configuration of the Email Listener service for your Creatio instance based on the average flow of emails (both incoming and outgoing) that the company mailboxes handle per second.

For example, if your company uses a single support mailbox whose email flow is 4, the recommended configuration includes 15 EL Worker replicas, 4 EL API replicas, and RabbitMQ service.

<table>
<tr><th>Number of mailboxes</th><th>Average email flow per second</th><th>Number of EL Worker replicas</th><th>Number of EL API replicas</th><th>RabbitMQ</th></tr>
<tr><td rowspan="3">1-150</td><td><1</td><td>0</td><td rowspan="3">4</td><td rowspan="6">Required</td></tr>
<tr><td>1-5</td><td>15</td></tr>
<tr><td>>5</td><td>25</td></tr>
<tr><td rowspan="3">>150</td><td><1</td><td>2</td><td rowspan="3">1 replica per each 30 mailboxes</td></tr>
<tr><td>1-5</td><td>15</td></tr>
<tr><td>>5</td><td>25</td></tr>
</table>

Note. The number of active EL Worker replicas directly affects the email handling speed. The email flow in production fluctuates, therefore certain EL Worker replicas might stand idle for some time. The article provides recommended configuration parameters, but you can use fewer replicas than the table specifies. In this case, the service will take longer to handle the emails during peak load. Optimize the ratio between the email handling speed and the number of resources utilized according to business requirements.

Component replica system requirements

Component	vCPU	RAM
EL Worker	0.1	1.1 GB
EL API	0.150	850 MB
Redis	0.5	3 GB
Rabbit MQ	0.5	4 GB

Note. The values in the table are recommended, the actual resource consumption might vary by the service use case. We recommend monitoring the CPU and memory resources on the deployed services to

optimize the available limits.

Email Listener deployment methods

We recommend **using the Kubernetes orchestrator and Helm package manager** to deploy the service. Read more >>>

You can also **use Docker** to speed up the deployment in the development environment. Read more >>>

Deploy the synchronization service via Kubernetes

Deploy the synchronization service using the RabbitMQ programmable message broker.

Take the following steps to deploy the service:

1. Set up the target environment:
   a. **Kubernetes cluster**. Learn more about setting up and managing the cluster in the Kubernetes documentation.
   b. **Helm package manager**. Learn more about installing the package manager in the Helm documentation.
2. **Install Redis**. Learn more about installing Redis using Helm on the GitHub website.

   Example of a command that installs Redis

   ```
   helm install --namespace default --set auth.enabled=true --set auth.password=password --set s
   ```

   Where:

   default is the namespace to install Redis.

   redis is an arbitrary name for the Redis instance.
3. **Install the RabbitMQ queue broker**. Use the standard bitnami/rabbitmq helm package in the Exchange Listener namespace with the recommended RabbitMQ parameters. Learn more: bitnami/rabbitmq (GitHub).

   Note. To specify the RabbitMQ memory limit, use the rabitmqMemoryHighWatermark option. The option is enabled by default in the bitnami/rabbitmq helm package. Calculate the limit using the formula below:
   `## rabbitmqMemoryHighWatermark = 0,4 * resources.limits.memory`

   Example that installs RabbitMQ with the minimum set of arguments

   ```
   helm install --atomic --namespace exchange-listener --set resources.limits.cpu=500m --set res
   ```

   Create a virtual host and Email Listener user in the installed RabbitMQ instance:

   a. Connect to RabbitMQ and run the following command:

```
kubectl exec test-rabbit-rabbitmq-0 -n exchange-listener --stdin --tty shell-demo -- /bin/l
```

b. Create a virtual host:

```
rabbitmqctl add_vhost ExchangeListener
```

c. Create a user and specify the password. For example, “creatio:”

```
rabbitmqctl add_user creatio
```

d. Set up the user permissions to the virtual host:

```
rabbitmqctl set_permissions --vhost ExchangeListener creatio ".*" ".*" ".*"
```

e. Install the Email Listener module. To install the module, download the helm package. View the available parameters of the helm package in the table below.

Attention. You need **repository access** to set up the current version of the Email Listener service. Contact Creatio support to verify your license and gain access to the repository.

For newer **Kubernetes versions**, specify the API version by adding the following parameter: `--set apiVersion=apps/v1`

Example of a command that uses the address and relative path

```
helm upgrade -i

--set ingress.enabled=true

--set ingress.path=<listener_path>

--set ApiUrl=kubernetes

--set apiVersion=apps/v1

--set-string Redis.Connection="<redis_host>\,password=<password>"

--namespace <namespace_name>

--set WorkerReplicaCount=2

--set ReplicaCount=2
```

```
--set RabbitMQ.ExchangeName=NewExchange;

--set RabbitMQ.QueueName=NewQueue;

--set RabbitMQ.Host=test-rabbit-rabbitmq;

--set RabbitMQ.HostApi=test-rabbit-rabbitmq;

--set RabbitMQ.HttpPort=15672;

--set RabbitMQ.AmqpPort=5672;

--set RabbitMQ.VirtualHost=ExchangeListener;

 --set RabbitMQ.Login=creatio;

--set RabbitMQ.Password=creatio; exchangelistener ./home/creatio/exchangelistener-1.0.13.tg

--set service.type=NodePort

--set service.nodePort=port
```

Where:

<redis_host> is the Redis server address.

<kubernetes_url> is the Kubernetes URL or IP address.

ReplicaCount is the number of EL API replicas based on the number of mailboxes and average email flow for your company. View the calculation table above.

WorkerReplicaCount is the number of EL Worker replicas based on the number of mailboxes and average email flow for your company. View the calculation table above.

To set up an HTTPS connection, deploy the service with Ingress and a valid SSL certificate, as well as specify HTTPS in the <kubernetes_url> Email Listener service address.

To check the availability, execute the query as specified in the Fig. 1.

```
<kubernetes_url>/<listener_path>/api/listeners/status
```

Example of a command that uses Node IP and port address:

```
helm upgrade -i exchangelistener ./home/creatio/exchangelistener-1.0.13.tgz --namespace de
```

Fig. 1 Email Listener service response

```
{
  "ServiceStatus": "Started",
  "version": "0.5.0",
  "connections": {
    "657b3ea8-477f-419c-a07a-4d4cc2158fc5": {
      "SenderEmailAddress": "[redacted]",
      "BpmUser": "[redacted]",
      "BpmEndpoint": "https://[redacted]/0/ServiceModel/ExchangeListenerService.svc/NewEmail",
      "State": "exists",
      "Id": "657b3ea8-477f-419c-a07a-4d4cc2158fc5",
      "RedisKey": "Subscription_657b3ea8-477f-419c-a07a-4d4cc2158fc5_exchangelistener-api-2",
      "UseFullEmail": true
    },
    "332bcae2-0530-4636-bc09-50a594389f53": {
      "SenderEmailAddress": "[redacted]",
      "BpmUser": "[redacted]",
      "BpmEndpoint": "https://[redacted]/0/ServiceModel/ExchangeListenerService.svc/NewEmail",
      "State": "exists",
      "Id": "332bcae2-0530-4636-bc09-50a594389f53",
      "RedisKey": "Subscription_332bcae2-0530-4636-bc09-50a594389f53_exchangelistener-api-1",
      "UseFullEmail": true
    },
```


Available parameters of the Email Listener helm package

Parameter	Parameter description	Default value
replicaCount	Number of StatefulSet handlers.	2
service.type	Service type. Learn more about the Kubernetes service types in the Kubernetes documentation.	ClusterIP
service.nodePort	If the service.type parameter equals NodePort, specify the external service port in this parameter. Learn more about the NodePort type in the Kubernetes documentation.	
env.host	Host address for Redis.	
env.port	Host port for Redis.	6379
env.base	Database number for Redis.	0.
ingress.enabled	Use address overriding via ingress.	false
ApiUrl	Service address if ingress.enabled=true	
ingress.path	Relative service path.	
log4Net.level	Default logging level.	Info

Use the system requirements calculator to check the server requirements.

Deploy the synchronization service in Docker

To set up the service, use a server (computer or virtual machine) that has Linux or Windows OS installed.

Attention. We recommend deploying the synchronization service in Docker only to the development environment. This method provides a high deployment speed, but does not enable compliance with the requirements of the product environment, namely: function fault tolerance, scaling for the handling of large request volumes, and a unified approach to component management that uses the container orchestration. For the product environment, we strongly recommend using the Kubernetes orchestrator and Helm package manager.

You need **repository access** to set up the current version of the Email Listener service. Contact Creatio support to verify your license and gain access to the repository.

Take the following steps to deploy the service:

a. Set up the Docker container platform first.

   To install Docker Desktop on Windows Server, follow special instructions on Microsoft website.

   To install Docker on Linux, follow the guide in the Docker documentation. To check the installed Docker version, run the following command:

   ```
   docker --version.
   ```

   You can install Docker components using the Docker-Compose instruction file. Learn more about installing Docker-Compose in the Docker documentation.

b. Install and run Email Listener:

   a. Open the directory to deploy Email Listener on the server dedicated for the service.

   b. Download and unpack the archive that contains the setup files to the directory. Download the archive.

   c. Open the / Creatio Email Listener component directory and run the following command:

      ```
      docker-compose up -d
      ```

      The command might take up to several minutes to complete.

c. Make sure the logs contain no errors by running the following command: docker logs ListenerAPI.

d. Check whether the deployment is complete by opening the http://localhost:10000/ URL, where localhost is the URL of the Email Listener server.

Set up the Email Listener service in Creatio

a. Make sure the ExchangeListenerService anonymous service is available at [*Creatio application address*]/0/ServiceModel/ExchangeListenerService.svc (Fig. 2).

Fig. 2 ExchangeListenerService response

Service

This is a Windows© Communication Foundation service.

Metadata publishing for this service is currently disabled.

If you have access to the service, you can enable metadata publishing by completing the following steps to modify your web or application configuration file:

1. Create the following service behavior configuration, or add the <serviceMetadata> element to an existing service behavior configuration:

```
<behaviors>
    <serviceBehaviors>
        <behavior name="MyServiceTypeBehaviors" >
            <serviceMetadata httpGetEnabled="true" />
        </behavior>
    </serviceBehaviors>
</behaviors>
```

2. Add the behavior configuration to the service:

```
<service name="MyNamespace.MyServiceType" behaviorConfiguration="MyServiceTypeBehaviors" >
```

Note: the service name must match the configuration name for the service implementation.

3. Add the following endpoint to your service configuration:

```
<endpoint contract="IMetadataExchange" binding="mexHttpBinding" address="mex" />
```

Note: your service must have an http base address to add this endpoint.

The following is an example service configuration file with metadata publishing enabled:

```
<configuration>
    <system.serviceModel>

        <services>
            <!-- Note: the service name must match the configuration name for the service implementation. -->
            <service name="MyNamespace.MyServiceType" behaviorConfiguration="MyServiceTypeBehaviors" >
                <!-- Add the following endpoint.  -->
                <!-- Note: your service must have an http base address to add this endpoint. -->
                <endpoint contract="IMetadataExchange" binding="mexHttpBinding" address="mex" />
            </service>
        </services>

        <behaviors>
            <serviceBehaviors>
                <behavior name="MyServiceTypeBehaviors" >
                    <!-- Add the following element to your service behavior configuration. -->
                    <serviceMetadata httpGetEnabled="true" />
                </behavior>
            </serviceBehaviors>
        </behaviors>

    </system.serviceModel>
</configuration>
```

For more information on publishing metadata please see the following documentation: http://go.microsoft.com/fwlink/?LinkId=65455.

b. Set the needed system setting values. To do this:

   a. Open the System Designer, e. g., by clicking .

   b. Click “System settings” in the “System setup” block.

   c. Set the values of the following system settings:

      “**ExchangeListenerServiceUri**” (the “ExchangeListenerServiceUri” code). The format of the system setting: [*the service address used at installation*]/api/listeners.

      “**Creatio exchange events endpoint URL**” (the “BpmonlineExchangeEventsEndpointUrl” code). The format of the system setting value: [*the anonymous ExchangeListenerService address*]/NewEmail. For example, https://mycreatio.com/0/ServiceModel/ExchangeListenerService.svc/NewEmail.

Email Listener service diagnostics

The Email Listener service diagnostics page provides tools for troubleshooting the service.

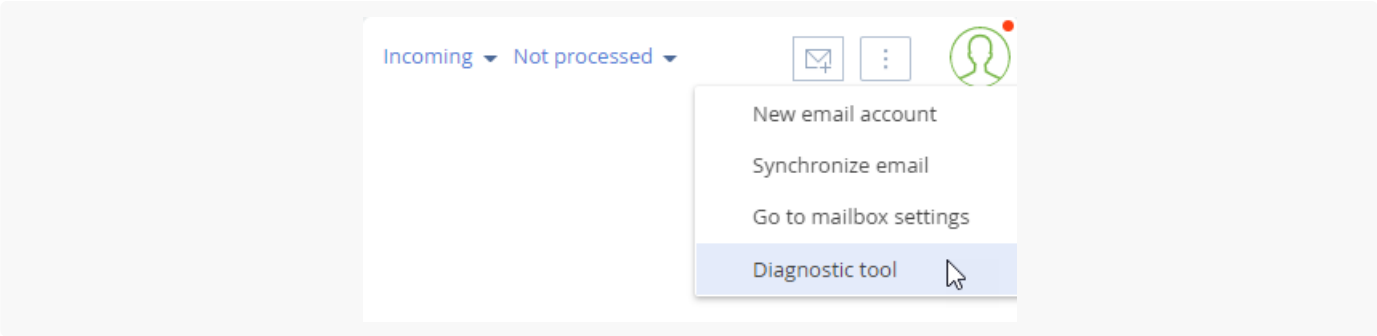
Use the service page:

- to check if the features essential for Exchange Listener are enabled
- to verify the service availability
- to receive subscription information
- to validate the “ExchangeListenerServiceUri” system setting
- to check the health of a mailbox
- to check if the microservice can connect to the Creatio website

You can open the Email Listener service diagnostics page in several ways:

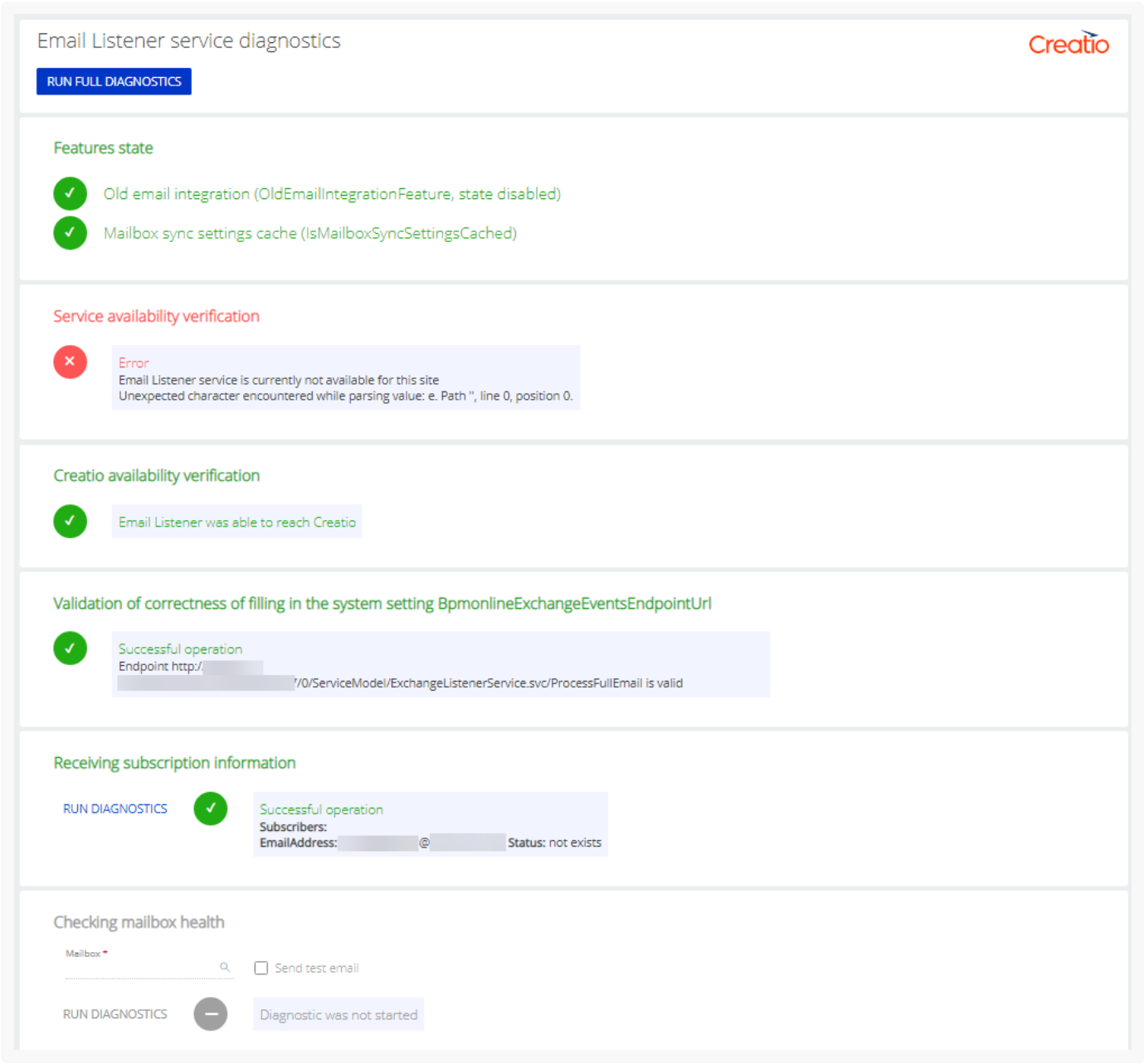
- Use the menu on the Email tab of the communication panel (Fig. 3).
- Use the page of the configured mail services.
- Add the “/0/ClientApp/#/IntegrationDiagnostics/” string to the URL of your Creatio website in the browser address bar and press Enter. For example, `http://mycreatio.com/0/ClientApp/#/IntegrationDiagnostics/ExchangeListener`.

Fig. 3 Open the Email Listener service diagnostics



The diagnostics page contains several readout blocks and diagnostics controls (Fig. 4). By default, most of the readout blocks do not display diagnostics data unless you click [*Run diagnostics*] in that block.

Fig. 4 Email Listener service diagnostics



Feature state	This readout block runs diagnostics automatically on page load. Checks if Email Listener features are enabled in your Creatio application: • ExchangeListenerEnabled • EmailIntegrationV2 • SendEmailsV2 Learn more about enabling features in the developer documentation: Feature Toggle mechanism.
Service availability verification	Checks if the Email Listener service is accessible from your Creatio application.
Receiving subscription information	Checks the connection to the remote server.
Validation of the “ExchangeListenerServiceUri” system setting	Checks if the Exchange Listener service endpoint specified in the “ExchangeListenerServiceUri” system setting is valid.
Checking mailbox health	Checks the operation of Microsoft Exchange mailboxes. Select the [*Send test email*] checkbox to send a test email to the specified address when clicking the [*Run diagnostics*] link.