Reporting Suite

Reporting

Minimum Requirements, Installation and Management Guide

Version 5.6.0

1/22/2026

Enghouse Interactive

Contents

About this Guide

Audience

This Guide is intended for the Administrator of Reporting. The Administrator must have a sufficient access level and needs to possess intermediate computer using skills in order to be able to use this document.

New in this release

No new information was added in this version of the document.

Guide conventions

This Guide uses the following text formats and notation conventions.

Text format

Bold text indicates a button, field, link, option name, or similar function requiring an action.

Italicized text indicates new terms, directory paths, or references to external documents.

`Text in this font` indicates code.

Notes and cautions

Icons used throughout this Guide identify additional details or special conditions.

> **Note**
> Provides additional information or describes special circumstances.

> **Caution**
> Warns of user actions that may cause system failure or irreversible conditions.

> **Stop**
> Describes actions that you should only perform under the supervision of Enghouse Interactive Customer Support.

Legal disclaimer

This document is governed by the terms of the software license agreement and applicable contract (including addendums) entered into with Enghouse.

Contact information

To submit comments or questions about the content in this Guide, please open a case in Support.

Installation - Docker-compose

Reporting is a web application that allows multiple Enghouse products to provide reporting and data visualization features to customers. Starting from version 4.0, Reporting can be installed in cloud environments and used in multi-tenant mode. In this installation configuration, it is recommended to provide automatic monitoring and alerting of the system health check.

The following topics cover the topics relevant for the installation and management of Reporting in a Docker-compose environment.

Product requirements

Required OS

To run Reporting V4.X or newer, you can use a VM template (provided by the BI team) to create a Linux server, with Ubuntu 20.04 as the OS.

Supported browsers

- Google Chrome
- Firefox 78.0.2 or higher
- Edge based on Chromium (79 or higher)

Required servers

Reporting V4.X or later can be installed in multiple configurations, from small single node installations to multi-node cluster installations. The appropriate number of VMs and their sizing have to be defined based on the expected system load. The following table can be used as a guideline for sizing the VMs:

		CC Agents (10 agents => 1 BI viewer)							
		200				**500**			
Installation type	**Server type**	**Num**	**RAM**	**CPU**	**DISK**	**Num**	**RAM**	**CPU**	**DISK**
Single Server	Reporting	1	8	8	120	1	12	10	120
Single Server HA	Reporting	2	8	8	120	2	12	10	120
Cluster	Nginx	1	2	2	30	1	2	2	30
	Engine	1	6	4	80	1	6	6	80
	Application DB	1	6	4	80	1	6	4	80
Cluster HA	Nginx	2	2	2	30	2	2	2	30
	Engine	2	6	4	80	2	6	6	80
	Application DB	2	6	4	100	2	6	4	100

		CC Agents (10 agents => 1 BI viewer)							
		1000				2000			
Installation type	**Server type**	**Num**	**RAM**	**CPU**	**DISK**	**Num**	**RAM**	**CPU**	**DISK**
Single Server	Reporting	1	18	16	140	n.d.	n.d.	n.d.	n.d.
Single Server HA	Reporting	2	18	16	140	n.d.	n.d.	n.d.	n.d.
Cluster	Nginx	1	2	2	30	1	2	2	30
	Engine	1	10	10	100	1	16	16	100
	Application DB	1	8	6	100	1	8	6	100
Cluster HA	Nginx	2	2	2	30	2	2	2	30
	Engine	2	10	10	100	2	16	16	100
	Application DB	2	8	6	100	2	8	6	100

Additional required elements In order to successfully install, run and use Reporting, the following elements need to be installed on your Ubuntu server (instructions described in *System initialization on page 9*):

- Docker and Docker Engine
- Docker Compose

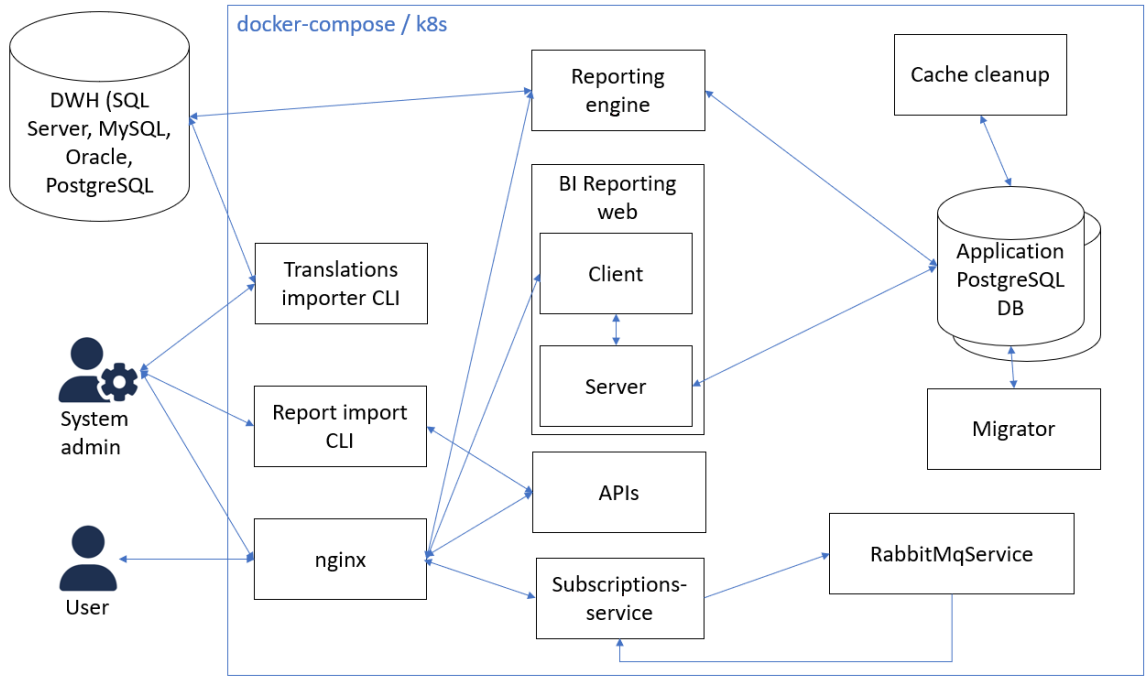
> **Note**
> Monitoring tools can require additional components.

Internet Access

Internet access is needed to install Reporting requirements (docker, docker-compose).

Architecture

Below you can find a schematic overview of the Reporting system architecture:



The following table provides a mapping between the logical module and the service/deployment that expose it in deployments based on Docker-Compose and Kubernetes.

Logical service	Docker-compose service	Kubernetes deployment	Description
nginx	nginx.service	nginx	**nginx** is used to proxy connection from clients to the proper service; nginx is also used to provide HTTPS access and manage certificates.
Translations importer CLI	translations-importer-cli.service	translations-import-tool	**Translations importer** is a CLI tool that allow system administrators to import standard and custom translations in the DHW.
Report importer CLI	std-report-import-tool	std-reports-import-tool	**Report importer** is a CLI

Logical service	Docker-compose service	Kubernetes deployment	Description
			tool that allow system administrators to import standard and custom reports in ERS, making them available to all tenants.
Reporting engine	reporting.service	reporting	**Reporting engine** is the service that manages the datamodel, report and dashboard design and visualization. It is the module that exposes the stimulsoft features.
BI Reporting web	web.service	web	**BI Reporting web** is the web application that exposes all the UI for accessing reports and dashboards, share and schedule assets, configure and manage the system, including uses and groups.
APIs	api.service	api	**APIs** is the service that exposes APIs useful for integrations of other products with ERS, as well as APIs internally used by other services.
Subscriptions-service	subscriptions.service	subscriptions	**Subscriptions service** is the component of ERS in charge of distributing scheduled reports and dashboards to users via email and other supported channels.
Cache cleanup	cachecleanup.service	cachecleanup	**Cache cleanup** is a sort of “garbage collector” that cleans up the internal cache keeping recently used data and removing whatever is considered obsolete and no longer

Logical service	Docker-compose service	Kubernetes deployment	Description
			useful.
Application PostgreSQL DB	db.service, pg-0, pg-1	ers-pgoperator, bireporting-pgdb-pgbouncer	ERS uses an internal instance of PostgreSQL to store configurations, users, assets and everything that needs to be persistant. It doesn't include the DWH or any other datasources used in reports. Postgres is replicated in master-slave replica mode, based on Percona PG Operator for kubernetes and bitnami pgpool for docker compose.
Migrator	migrator.service	migrator (*job*)	Migrator is a tool in charge of upgrading the application database to the latest version; it's run once at startup of the application with docker compose deployment, while it's created as a single shot job during helm installation in case of kubernetes deployment.
RabbitMqService	rabbitmq.service	ers-rabbitmq-cluster-operator, ers-rabbitmq-messaging-topology-operator	**RabbitMQ** is a messages queue manager used internally by ERS to allow services to communicate. It's mainly used by the **web** and **subscriptions** services to properly manage notification of changes in instances where multiple replicas of each service are enabled.

System initialization

> **Note**
> The following information refers to Reporting Version 4.X and newer.

Server installation

The new VM on the Linux server needs to be installed with Ubuntu.

Network configuration

Once you've created a new VM, you need to configure the correct IP address and server name.

To set up a new virtual machine, do the following:

1. Before starting the server, verify that the network is not connected in any way (including on startup).
2. Startup the VM.
3. Connect to the VM with a remote or web console.
4. Edit the following files (using any file editor you typically use on Linux) to configure your IP address and server name:

- `/etc/hosts`
  - Replace "enghousebi" with your-server-name and "127.0.0.1" with your new IP; do not use special characters.
  - Then add a new row at the end of the file with your server IP and server name, e.g. "1.2.3.4 your-server-name"
- `/etc/hostname`
  - Replace"enghousebi" with your server name; do not use special characters
- `/etc/netplan/00-installer-config.yaml`
  - The file needs to contain the information described below, relevant for your system. You have to review all the elements included in the <> brackets (as seen in the example above) with your network-specific values.

```
network:
ethernets:
ens33:
dhcp4: false
addresses:
- <server_ip>/<network_mask>
gateway4:<gateway_ip>
nameservers:
addresses:
- <nameserver1_ip>
```

```
- <nameserver2_ip>
version: 2
```

5. Restart the server with the `init 6` command.
6. Connect the server to the network and allow it to connect at power on/startup.

Reporting Requirements

To install Reporting, Docker and Docker Compose need to be up and running on all Reporting servers.

> **Caution**
>
> **Internet access** is required to install Reporting requirements through the procedure described in this section.

To install Docker on your Ubuntu server, run the following procedure:

- Set up the repository:
  - Update the apt package index and install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get update

sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release
```

  - Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

  - Use the following command to set up the stable repository:

```
echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-
archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

- To install the Docker Engine:
  - Update the apt package index, and install the latest version of Docker Engine and container:

```
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- To install Docker Compose on your Ubuntu server, run the following procedure:
  - Download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Getting an account for docker registry

> **Note**
> This step is only required for online installations.

In order to get grants to download docker images from the registry where Reporting images are stored, you need to create a personal account. A personal account is required for each cloud installation or on-prem customer.

The account request should be opened by Enghouse employees only, opening a JIRA ticket at IP-Cloud-Enablement, and setting the Component attribute to Artifacts Registry.

When a new account is requested, the following info is required:

- Customer official name
- Deployment contact email address, must be within the company
- Product(s) being deployed

The generated login and temporary password will be sent solely to the provided email address, and the ticket will be closed when this is completed so that Enghouse individuals are notified upon completion.

> **Caution**
> The account creation procedure is not real-time and can take some time to be completed; it is highly recommended to run this procedure a few days before starting the installation procedure to avoid delays in the next steps.

Authenticating to the docker registry

> **Note**
> This step is only required for online installations.

To authenticate to the docker registry, you just have to run the following command from a SSH session on the Reporting server:

```
sudo docker login registry.eptica.com
```

When requested, provide the credentials generated following the procedure in the previous section. The system will automatically save your account info for any future access to the docker registry.

If you want to log out of the docker registry, run the following command:

```
sudo docker logout registry.eptica.com
```

Reporting Installation

> **Note**
>
> The following information refers to Reporting Version 4.X and newer.

Once all the Reporting servers are properly installed following the instructions from previous sections, you need to configure the Reporting infrastructure and install the components. To install Reporting on your Ubuntu server, run the following procedure:

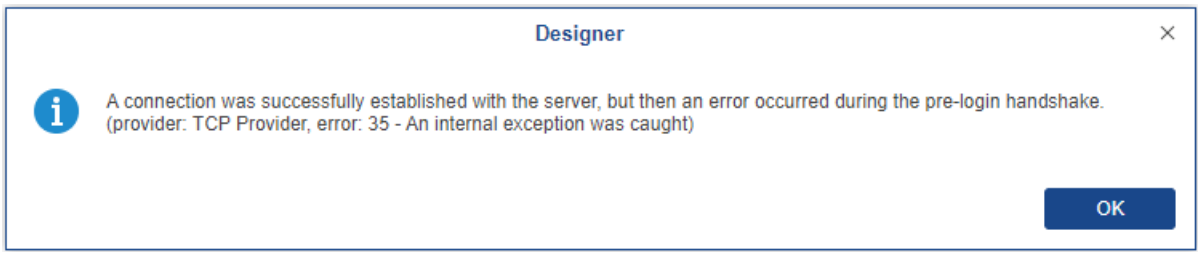
1. Look for the version specific Reporting files on Sharepoint and then copy the Reporting files to the server; the default installation folder is `/home/enghouse/bireporting`.

> **Caution**
>
> Copy the single uncompressed file to the server and then unpack it. Otherwise, you will have file and/or folder permission issues.

2. To create the default folder, SSH into the server and run: `mkdir -p /home/enghouse/bireporting`
3. Navigate to the folder where you copied the Reporting install files: `cd /home/enghouse/bireporting`
4. Rename *.env.example* to *.env*: `mv .env.example .env`
5. Edit the `.env` file using a text editor: `vi .env`
6. The file should have the following environment variables set for you to edit (values are examples and might not represent values that you should actually use):
    - `DB_HOST` should always be `db.service` (name of the Postgres database container in Docker Compose).
    - When composing for the first time, the `RUN_MIGRATIONS` variable value should be 1, that way the database is created and migrated.
        - In case of a future update, to disable the running of new migrations in order to prevent the database from breaking before performing the backup, you can set RUN_MIGRATIONS to 0.
    - `USE_DESIGNER_DB_CACHE=false` and `USE_VIEWER_DB_CACHE=false` can be used to disable database cache for reporting.service. By default database cache is enabed. Database cache is used to run reporting.service as a stateless services which can be scaled horizontally. If database cache is disabled, then in-memory cache is used, and service cannot be scaled. This is not a problem for deployments with only single instance of reporting.service.
    - `LOG_TO_FILE=true` can be used to enable storing logs in rolling files in addition to console logs which are always generated. This flag is global, and its value applies to all services. By default this option is disabled.
    - `SSO_TIMEOUT` can be used to configure timeout for requests sent to identity server in case when SSO through OIDC is enabled in the app. Default timeout is 3500.
    - `SSO_MAX_RETRIES` limits the number of retries for requests sent to SSO identity server in case of response timeout (applicable when SSO through OIDC is enabled). Default value is 5.
    - `SSO_WAIT_BEFORE_RETRY_MS` specifies wait period (in milliseconds) between retry requests sent to SSO identity server (applicable when SSO through OIDC is enabled). Default value is 1000.

It is possible to override the default settings of MinProtocol and `CipherString` in `/etc/ssl/openssl.cnf` within `reporting.service`.

An example of error that could occur connecting to a database that can be fixed by reviewing `MinProtocol` and `CipherString` settings could be such:



The default settings are:

- `MinProtocol = TLSv1.2`
- `CipherString = DEFAULT@SECLEVEL=2`

To override them, add the following environment variables in the .env file:

- `MIN_PROTOCOL`
- `CIPHER_STRING`

As an example, it is possible to set TLSv1 as follows: `MIN_PROTOCOL="TLSv1"`

or set the default security level to 1, as follows: `CIPHER_STRING="DEFAULT@SECLEVEL=1"`

To continue with the installation, move on with these steps:

7. Provide SSL certificates:

- Create a new sub-directory called certs: `mkdir certs`
- Put `certificate.crt` into the `certs` folder, which is the SSL certificate to be used for the application.
- Put `private.key` into the `certs` folder, which is the key of the SSL certificate.

> **Caution**
> Make sure you do not rename the certificate files, as the exact name and path specified are expected.

> **Note**
> It is possible to update certificates just replacing the old ones; when replaced, it is required to restart Reporting Suite to make changes effective.

Note

If you need to create self-signed certificates for testing or dev environments, run the following command after creating the certs folder:

`openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout certs/private.key -out certs/certificate.crt`

Then provide the requested information to create self-signed certificates. Keep in mind that SSO doesn't allow self-signed certificates.

If you are not using signed certificates, it may be necessary to disable certificate validation for the web.service component; it can be done by adding the option `NODE_TLS_REJECT_UNAUTHORIZED: 0` to the web.service environment variables in the `docker-compose.yml` file, as shown in the following image:

Updating rabbitmq default passwords

1. When in folder `home/enghouse/bireporting` check the rabbitmq image version with the command: `cat docker-compose.yml | grep image:.*rabbitmq`
2. The result will be e.g., `image: rabbitmq:3.12.6-management` which means the image used is `rabbitmq:3.12.6-management`.
3. After that run the helper container with the specified image with the command: `sudo docker run --name helper_rabbitmq --detach rabbitmq:3.12.6-management`
4. Once the container is started, connect to it with this command: `sudo docker exec -it helper_rabbitmq bash`
5. In the container execute the command for hashing passwords. Here instead of "examplepass" you will enter actual passwords you want to use for rabbitmq admin and/or for rabbitmq service user: `rabbitmqctl hash_password examplepass`
6. Copy the resulting hashes, store them outside the container. Exit the container with Ctrl+C. Execute the following commands to stop and remove the container:

   ```
   sudo docker stop helper_rabbitmq

   sudo docker rm helper_rabbitmq
   ```

7. To apply the new hashes for users, the file `home/enghouse/bireporting/rabbitmq/definitions.json` needs to be updated in the section "users". For users admin and/or subsuser default password hashes need to be replaced with the generated password hashes in the "password_hash" field for the user.
8. Also in `home/enghouse/bireporting/.env` he RABBITMQ_URL property needs to be changed to use the new plaintext password of the subsuser (instead of the default one), e.g., if new password is

`newsubsuserpassword` the URL will be: `RABBITMQ_URL-L="amqp://subsuser:newsubsuserpassword@rabbitmq.service/subshost"`

9. After the application is up, the rabbitmq management interface can be accessed at `https://host:-port/rabbitmq` with the user admin and with the password set for the admin user.

Loading docker images without internet access

> **Note**
> This step is only required for offline installations.

Starting from Reporting 4.4, it is possible to manually upload Reporting images to Reporting servers. This makes the installation procedure a bit more complex, but allows offline installations and upgrades.

To manually load the docker images, upload the "images" folder provided with the Reporting package to the Reporting server into the `/home/enghouse/` folder. The following `tar.gz` files are available in `/home/enghouse/images`:

- api.tar.gz
- enghouse-bi-dwh-liquibase.tar.gz
- migrator.tar.gz
- reporting-engine.tar.gz
- reporting-engine-cachecleanup.tar.gz
- std-report-import-tool.tar.gz
- translations-importer-cli.tar.gz
- web.tar.gz
- subscriptionsservice.tar.gz

> **Caution**
> The Images path should be in the `..\images` relative path, starting from the Reporting folder. If you are not using the default folders structure provided in previous steps, adjust the Images path accordingly.

Once the images files are ready, enter the Reporting folder and run the following command:

```
./importLocalImages.sh
```

and wait for execution completion.

Starting up Reporting

> **Caution**
> It is recommended to never change the installation folder name after first startup, as the folder name is used to define docker volumes; a change in the installation folder name will result in losing the application database content, which means a reset of all the application settings, users, groups, data models and reports.

To startup Reporting, run the following procedure:

- Start the containers: `sudo docker-compose up -d`
- To check the status of Docker Composed containers, run: `sudo docker-compose ps`
- To access containers' logs, run: `sudo docker-compose logs --tail="all" --follow`

Caution

It is possible to retrieve the initial password for the Enterprise Reporting System **system.admin** user right after the first Reporting startup with:

```
echo -e $(sudo docker-compose logs | grep "Password is")
```

Docker Compose logs are available until the next restart. In case of a restart, the password can be still retrieved from persistent logs available in the following folder:

```
/var/lib/docker/volumes/bireporting_logs/_data
```

This is done by running:

```
sudo -i
cd /var/lib/docker/volumes/bireporting_logs/_data
grep "Password is" BI-Reporting-api-*.log
```

You will need `sudo` grants to access the logs folder and the folder name may differ if a custom installation folder was used for Reporting.

Note

The first run can fail due to the database initialization process; in that case, some of the services will result in a "restarting" status when using the "docker-compose ps" command. To resolve this, restart the environment with the `down` and then `up` commands:

```
sudo docker-compose down
sudo docker-compose up -d
```

Importing

> **Note**
> The following information refers to Reporting Version 4.X and newer.

> **Caution**
> It is recommended to never change the installation folder name during upgrades, as the folder name is used to define Docker volumes; a change in the installation folder name will result in losing the application database content, which entails a reset of all the application settings, users, groups, data model and reports.

Importing standard data models, reports and views

When standard elements (data models, reports and views) are used across all tenants, the import procedure can be run after Reporting has been installed.

Once you have installed Reporting and it is up and running, access Reporting with an SSH client and follow these steps:

1. Enter the folder where Reporting is installed (by default, `bireporting`): `cd /home/eng-house/bireporting`
2. Access the `std-report-import-tool` container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
3. Run the upgrade process: `node std-report-import-tool.js`
4. You will be asked to enter the user password for system.admin user; enter the password and press enter.
5. Check the execution logs and then exit: `exit`

Importing integration-specific data models, reports and views

When integration-specific elements (data models, reports and views) are used across all tenants (e.g., product-specific elements), the import procedure can be run after Reporting has been installed.

Once you have installed Enghouse Reporting and it is up and running, access it with an SSH client and follow these steps:

1. Enter the folder where Enghouse Reporting is installed (by default, `bireporting`): `cd /home/eng-house/bireporting`
2. Access the std-report-import-tool container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
3. List the available integration folders: `ls reports/integrations`

4. Edit the settings.json file. This file contains default parameters for connecting Enghouse Reporting APIs and the folder that includes the report that you have to import: `vi settings/settings.json`
5. Replace the default value of the `reportsPath` parameter with the proper integration folder, e.g., `reports/integrations/CCaaS`
6. Run the upgrade process: `node std-report-import-tool.js`
7. You will be asked to enter the user password for system.admin user; enter the password and press enter.
8. Check the execution logs and then exit: `exit`

Import/Upgrade custom data models, reports and views

When custom elements (data models, reports and views) are used across all tenants (e.g., product-specific elements not officially shared with the Enghouse BI team), the import procedure can be run after Enghouse Reporting has been installed.

Once you have installed Enghouse Reporting and it is up and running, access it with an SSH client and follow these steps:

1. Create a new folder in `/home/enghouse/bireporting/reports` (e.g., `custom`):

```
cd /home/enghouse/bireporting/reports

mkdir custom
```

2. Upload your files into this folder.
3. Enter the folder where Enghouse Reporting is installed (by default, `bireporting`): `cd /home/enghouse/bireporting`
4. Access the `std-report-import-tool` container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
5. List the available integration folders: `ls reports`
    - you will see the folder created at step 1; folder /home/enghouse/bireporting/reports is mapped into the std-report-import-tool container.
6. Edit the `settings.json` file: `vi settings/settings.json`
    - This file contains default parameters for connecting Enghouse Reporting APIs and the folder that includes the report you want to import.
7. Replace the default value of the `reportsPath` parameter with the new folder folder, e.g., `reports/integrations/custom`
8. Run the upgrade process: `node std-report-import-tool.js`
9. You will be asked to enter the user password for system.admin user; enter the password and press enter.
10. Check the execution logs and then exit: `exit`

Upgrading

> **Note**
> The following information refers to Reporting Version 4.X and newer.

> **Caution**
> It is recommended to never change the installation folder name during upgrades, as the folder name is used to define Docker volumes; a change in the installation folder name will result in losing the application database content, which entails a reset of all the application settings, users, groups, data model and reports.

Upgrade requirements with internet access

If you don't have a docker registry account yet, request it as described in *Getting an account for docker registry on page 11*.

Once the registry account is available, if you are not yet logged in to the docker registry on your Reporting server, authenticate using the procedure described in *System initialization on page 9*.

Upgrade requirements without internet access

If your Reporting server doesn't have access to the internet, you have to manually load docker images to your server following the procedure described in *System initialization on page 9*.

Upgrading Reporting

To upgrade Reporting, run the following procedure:

1. Download the version-specific Reporting files on Sharepoint and then replace them in `/home/eng-house/bireporting`, optionally keeping a backup copy of the original ones and porting custom configurations from the old version to the new version.
2. Stop the Docker Compose container: `sudo docker-compose down`
3. Start the containers: `sudo docker-compose up -d`

Upgrading standard data models, reports and views

When standard elements (data models, reports and views) are used across all tenants, the upgrade procedure can be run after Reporting has been upgraded.

You must copy the `reports` folder from the distribution pack to `/home/enghouse/bireporting`, which is part of the Reporting upgrade procedure described above.

> **Caution**
>
> When replacing the files in `/home/enghouse/bireporting/reports` it is possible that some file has been renamed, in which case it should not be loaded into Reporting. It is recommended to delete old files before loading the new ones to avoid unpredictable results in the upgrade process.

Once you have upgraded Reporting and it is up and running, access Reporting with an SSH client and follow these steps:

1. Enter the folder where Reporting is installed (by default, `bireporting`): `cd /home/enghouse/bireporting`
2. Access the `std-report-import-tool` container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
3. Run the upgrade process: `node std-report-import-tool.js`
4. You will be asked to enter the user password for system.admin user; enter the password and press enter.
5. Check the execution logs and then exit: `exit`

Upgrading integration-specific data models, reports and views

When integration-specific elements (data models, reports and views) are used across all tenants (e.g., product-specific elements), the upgrade procedure can be run after Reporting has been upgraded.

It is important to copy the `reports` folder from the distribution pack to `/home/enghouse/bireporting`, which is part of the Reporting upgrade procedure described above. This folder contains standard elements, as well as elements provided to the Enghouse BI team as part of the official product integrations.

> **Caution**
>
> When replacing the files in `/home/enghouse/bireporting/reports` it is possible that some file has been renamed, in which case it should not be loaded into Enghouse Reporting. It is recommended to delete old files before loading the new ones to avoid unpredictable results in the upgrade process.

Once you have upgraded Enghouse Reporting and it is up and running, access it with an SSH client and follow these steps:

1. Enter the folder where Enghouse Reporting is installed (by default, `bireporting`): `cd /home/enghouse/bireporting`
2. Access the std-report-import-tool container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
3. List the available integration folders: `ls reports/integrations`
4. Edit the settings.json file. This file contains default parameters for connecting Enghouse Reporting APIs and the folder that includes the report that you have to import: `vi settings/settings.json`
5. Replace the default value of the `reportsPath` parameter with the proper integration folder, e.g., `reports/integrations/CCaaS`

6. Run the upgrade process: `node std-report-import-tool.js`
7. You will be asked to enter the user password for system.admin user; enter the password and press enter.
8. Check the execution logs and then exit: `exit`

Upgrade custom data models, reports and views

When custom elements (data models, reports and views) are used across all tenants (e.g., product-specific elements not officially shared with the Enghouse BI team), the upgrade procedure can be run after Enghouse Reporting has been upgraded.

You must copy your custom into the `reports` folder. This folder contains standard elements, as well as elements provided to the Enghouse BI team as part of the official product integrations.

Once you have upgraded Enghouse Reporting and it is up and running, access it with an SSH client and follow these steps:

1. Create a new folder in `/home/enghouse/bireporting/reports` (e.g., `custom`):

```
cd /home/enghouse/bireporting/reports
mkdir custom
```

2. Upload your files into this folder.
3. Enter the folder where Enghouse Reporting is installed (by default, `bireporting`): `cd /home/enghouse/bireporting`
4. Access the `std-report-import-tool` container with a shell prompt: `sudo docker-compose exec std-report-import-tool.service /bin/sh`
5. List the available integration folders: `ls reports`
   - you will see the folder created at step 1; folder /home/enghouse/bireporting/reports is mapped into the std-report-import-tool container.
6. Edit the `settings.json` file: `vi settings/settings.json`
   - This file contains default parameters for connecting Enghouse Reporting APIs and the folder that includes the report you want to import.
7. Replace the default value of the `reportsPath` parameter with the new folder folder, e.g., `reports/integrations/custom`
8. Run the upgrade process: `node std-report-import-tool.js`
9. You will be asked to enter the user password for system.admin user; enter the password and press enter.
10. Check the execution logs and then exit: `exit`

Upgrading the Enghouse BI DWH

When using standard reports, after a new version of Enghouse Reporting is released, the appropriate version of the Enghouse BI DWH needs to be used to guarantee that standard reports will work properly.

To upgrade the DWH, copy the `dwh` folder from the distribution pack to `/home/enghouse`.

Once the `dwh` folder is properly copied, access Enghouse Reporting with an SSH client and follow these steps:

1. Enter the dwh folder: `cd /home/enghouse/dwh`
2. Start the docker-compose environment: `sudo docker-compose up -d`
3. Access the `enghouse-bi-dwh-liquibase.service` container with a shell prompt: `sudo docker-compose exec enghouse-bi-dwh-liquibase.service /bin/bash`
4. Enter the liquibase folder: `cd liquibase`
5. Run the upgrade process; based on the DBMS you use for the DWH, you have to run the proper upgrade script:
   - SQL Server: `./sqlserverUpdate.sh`
   - MySQL: `./mysqlUpdate.sh`
   - PostgreSQL: `./postgresUpdate.sh`
   - Oracle: `./oracleUpdate.sh`
6. Depending on the specific DBMS, the syntax is slightly different. Run the script without any parameter to get the correct syntax. For example, to get the syntax for SQL Server, run `./sqlserverUpdate.sh`; the output is similar to the following:

```
     Syntax: ./sqlserverUpdate.sh dbServer dbUser dbPwd [dbName] [dbInstance] [dbPort]
Description: Runs the liquibase scripts to update SQL Server database
 Parameters:
          dbServer = Database server name or ip IP address
            dbUser = Database user
             dbPwd = Database password
            dbName = Database name (default: enghouse_bi_dwh)
        dbInstance = SQL Server instance (default: not specified)
            dbPort = Database port (default: not specified)
```

7. Based on the required syntax, run the script with the proper parameters to upgrade the DWH. If everything runs properly, you'll receive an output like the following:

```
Updating Enghouse BI DWH ...
Starting Liquibase at Wed, 11 Jan 2023 13:06:31 UTC (version 3.8.0 built at 2019-08-15T20:38:06Z)
Liquibase Community 3.8.0 by Datical
Liquibase: Update has been successful.
Enghouse BI DWH updated!
```

8. Check the execution output and then exit: `exit`
9. To free up the server resources, stop the docker-compose environment: `sudo docker-compose down`

> **Note**
>
> If you're using multiple tenants on the same Enghouse Reporting server, the above procedure has to be run on all the tenants' DWH.

Caution

- When entering the parameters for the shell script, keep in mind special character escaping or parameters quoting. Especially in passwords, you could have special characters like $, \, etc. Remember to escape them with the \ escape character or quote the parameter using single quotes. E.g., if your password is `Pa$$word`, you must escape the `$` character providing the password as `Pa\$\$word` or quote it like `'Pa$$word'`.
- If you are upgrading the DWH from version 4.4.0 or lower to version 4.4.1 or higher, on SQL Server 2016 or lower, you have to disable `clr strict security` if it is enabled. The commands to disable it are:

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'clr strict security', 0;
RECONFIGURE;
```

Backup and restore

> **Note**
> The following information refers to Reporting Version 4.X and newer.

Enghouse Reporting data and configurations are stored in an internal PostgreSQL database. In order to have a backup of all the application data, it is possible to manually run a backup (e.g. before a scheduled system upgrade) or schedule a periodic backup.

The application database contains the following data:

- Users (except SSO users)
- Groups
- Reports
  - Definition
  - Sharing
  - Schedules
  - Preferred filters
  - Bookmarks
- Data models
  - Definition
  - Sharing
- Configurations
  - SSO
  - Tenant

> **Note**
> Enghouse Reporting backup **does not** include the DWH or the call center data.

> **Note**
> Instances may occur where the backup/restore scripts don't have execution permission after the first installation or a software upgrade; in order to assign the execution permission, run the following command:
>
> `chmod +x /home/enghouse/bireporting/scripts/*.sh`
>
> Make sure to review the scripts path if Enghouse Reporting is not installed in the standard installation folder.

Enghouse Reporting Database Backup

The backup script is available in the scripts folder in the Enghouse Reporting installation folder; the default Enghouse Reporting installation folder is `/home/enghouse/bireporting`.

In multi-node environments, both manual and scheduled backups have to be run on the server that is running the application database. In order to verify if the server you are using is the correct one, run the following command:

```
sudo docker-compose ps db.service
```

If the server is running the application database, this command should return a row with the service status `Up`.

To manually run a backup, enter the Enghouse Reporting installation folder:

```
cd /home/enghouse/bireporting
```

and then run the backup script:

```
./scripts/BIReportingDBDUMP.sh
```

When the backup execution ends, a new backup file is available in the `backups/BIReporting` folder within the Enghouse Reporting installation folder. The backup files are named `bireportingdb_YYYY-MM-DD.bak`, where YYYY-MM-DD is the date when the backup was created.

> **Note**
>
> If the backup script is run multiple times in the same day, only the last version of the backup is kept. If you have to store multiple versions of backups for the same day, it is recommended to manually rename the old backup file before running a new backup.

> **Caution**
>
> The backup is stored in the `backups/BIReporting` folder within the Enghouse Reporting installation folder. It is recommended to configure a mechanism to pull those backups from this folder and store them in external storage; this pull mechanism should also define a cleanup policy, deleting copied files or deleting files older than few days.

To schedule a periodic backup of the Enghouse Reporting database, run the backup script using the `cron` feature of the hosting server.

The `cron` configuration can be done by creating a file named `bireportingbkp` in `/etc/cron.d` and editing it, for example with the vi editor:

```
sudo vi /etc/cron.d/bireportingbkp
```

then, add the following row:

```
0 1 * * * root /home/enghouse/bireporting/scripts/BIReportingDBDUMP.sh >>
/var/log/bireportingKBP.log 2>&1
```

The provided example runs a backup every day at 1.00 AM and writes the execution logs in `/var/log/bireportingKBP.log`. Read in the Linux documentation about the cron schedule and cron schedule syntax for more details.

Restoring the Enghouse Reporting Database

Enghouse Reporting provides a script that allows you to easily restore a database dump. A restore can be requested in case of a system recovery due to a server replacement or loss of information due to unexpected issues.

In order to restore the Enghouse Reporting database dump, you need to have a backup available in the `backups/BIReporting` folder; then, run the restore script specifying the name of the backup you want to restore, as follows:

```
./scripts/BIReportingDBRESTORE.sh bireportingdb_YYYY-MM-DD.bak
```

Make sure to replace YYYY-MM-DD with the date of the backup you have to restore.

Avoid running the restore procedure while a backup process is in progress. After a restore, it is recommended to restart Enghouse Reporting using the docker-compose `down` and then `up` procedure.

System monitoring

> **Note**
> The following information refers to Reporting Version 4.X and newer.

Server status and OS monitoring

You need to monitor the following resources on Enghouse Reporting servers:

- CPU
- Memory
- Disk

An alert should be raised as follows:

- CPU average load (last 5 minutes)
  - over 70% -> warning
  - over 85% -> critical
- Memory usage
  - over 80% -> warning
  - over 90% -> critical
- Disk usage
  - over 80% -> warning
  - over 90% -> critical

Docker containers monitoring

> **Note**
> This section applies only for docker-compose deployment, and will be obsolete and redefined after full migration to Kubernetes deployment.

Enghouse Reporting components run in Docker containers, managed with docker-compose.

Monitoring systems could provide specific tools for monitoring containers. Refer to the monitoring product documentation for detail on how to monitor docker containers.

The following table shows the required running containers and the servers where they are expected to run:

	Server Type			
Container name	**Enghouse Reporting**	**Nginx**	**Engine**	**Application DB**

api.service	X		X	
cachecleanup.service	X		X	
db.service	X			X
nginx.service	X	X		
rabbitmq.service	X		X	
reporting.service	X		X	
subscriptions.service	X		X	
web.service	X		X	

Note

The `bireporting` prefix is based on the installation folder of Reporting; the default installation folder is `/home/enghouse/bireporting`, as described in *System initialization on page 9*. If a different folder is used during installation, the container prefix needs to be changed as well, according to new folder name.

Enghouse Reporting Health Checks

Enghouse Reporting provides APIs that allow you to verify the healthy status of its components. It is useful to monitor them periodically to identify as soon as possible if there is an application issue. The APIs that should be monitored are listed in the following table:

Component	Method	Health check API URL
API service	GET	http(s)://<BIReportingHost>:<Port>/api/v2/HealthCheck
Reporting Engine	GET	http(s)://<BIReportingHost>:<Port>/ReportingEngine/HealthCheck
Reporting WebApp	GET	http(s)://<BIReportingHost>:<Port>/api/HealthCheck
Subscriptions Service	GET	http(s)://<BIReportingHost>:<Port>/subscriptions/HealthCheck

The expected result from the health check APIs is just a response code `200 OK`, with a JSON body like this:

```
{"status":"running"}
```

Refer to the monitoring product documentation for detail on how to monitor http APIs response.

Application database monitoring

Enghouse Reporting uses a PostgreSQL instance in order to store the internal configurations.

The PostgreSQL instance is provided in a Docker container named `bireporting_db.service_1`, already mentioned in *Docker containers monitoring on the previous page*.

Some monitoring systems provide specific tools for monitoring most common databases. Refer to the monitoring product documentation for detail on how to PostgreSQL databases.

Issue analysis

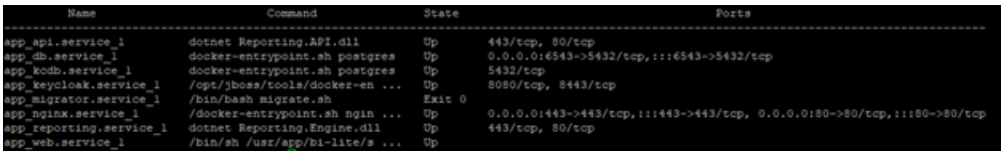
> **Note**
> The following information refers to Reporting Version 4.X and newer.

Docker-compose checks

When the system is not working properly, the first check that you need to do on the server is related to the Docker-compose containers status. To check it, log in to the server with an SSH client, access the Enghouse Reporting folder and run the following command:

```
sudo docker-compose ps
```

The expected result of the previous command should be similar to the following:



All the containers should be in the Up state, except for bireporting_migrator.service_1.

If one or more containers are not up and running, you need to investigate the reason for this in the application logs.

Application logs checks

Basics

When Enghouse Reporting is up and running, the docker-compose logs visualization can be used to access logs.

The basic command line to access logs is:

```
sudo docker-compose logs
```

Running the aforementioned command, all the logs starting from the last Enghouse Reporting startup are returned.

If you want to scroll and search within logs, you can use the `less` command, as follows:

```
sudo docker-compose logs | less -r
```

The “-r” option is needed to colorize output.

If you have to search specific strings within the logs, returning all the occurrences of that string, use the `grep` command; in the following example, all the occurrences of "error" (case insensitive) are searched:

```
sudo docker-compose logs | grep -i error
```

The Docker-compose `logs` command can also be used to focus on a specific container of Enghouse Reporting. It is enough to specify the container name after the `logs` command:

```
sudo docker-compose logs migrator.service
```

If needed, you can read logs for just one subset of the containers, specifying the containers list:

```
sudo docker-compose logs migrator.service web.service
```

This container selector can be combined with commands already described in this section, such as `less`:

```
sudo docker-compose logs migrator.service | less -r
```

Searching for issues

When the application is not working as expected, the recommended procedure is to search for key strings “error” and “exception” (case insensitive) within all the container logs, as follows:

```
sudo docker-compose logs | grep -i error
```

and:

```
sudo docker-compose logs | grep -i exception
```

If at least one row with “error” or “exception” is found, it is recommended to get the container name at the beginning of the row (removing the “_1” suffix at the end) and to deeply analyze the container logs with the `less` command:

```
sudo docker-compose logs migrator.service | less -r
```

Visualizing recent logs

As mentioned at the beginning of this section, the Docker-compose `logs` command returns, by default, all the logs since the last startup of Enghouse Reporting. If the system has been running for a long time, it could be useful to limit the logs to just the more recent rows. It is possible to specify the number of rows that you want to read counting from the end of the logs for each container with the following line:

```
sudo docker-compose logs --tail=100
```

This command can be combined with everything already described in this section, as in the following example, where the “error” string is searched for in the `migrator.service` container in the last 100 log rows:

```
sudo docker-compose logs --tail=100 migrator.service | grep -i error
```

Following new logs

When you are investigating an issue, and you have to visualize logs while you run a test on the application, it may be useful to follow new logs while they are generated. The Docker-compose `logs` command provides the "-f" option to "follow" new logs, as follows:

```
sudo docker-compose logs -f --tail=1
```

As in the example, usually the “follow” option is used in combination with a small tail value, to focus just on recent and new lines.

As for the other options, the “follow” option can be used in combination with other commands, such as the `grep` command, to just look at errors for a specific container:

```
sudo docker-compose logs -f --tail=1 api.service | grep -i error
```

To stop following the logs, press CTRL + C.

Persistent logs

Docker compose logs are not persistent; all the logs are deleted when the container is stopped. To access persistent logs, you can access the Enghouse Reporting server with an SSH client and get sudo grants:

```
sudo -i
```

then, access the logs volume:

```
cd /var/lib/docker/volumes/bireporting_logs/_data
```

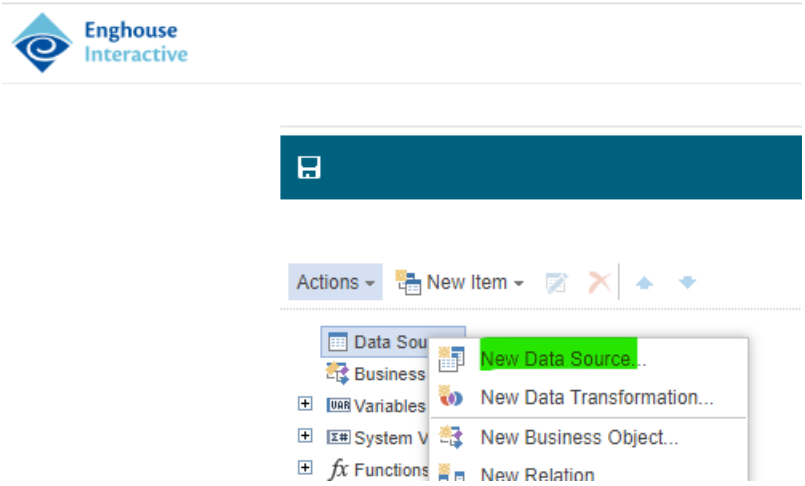
This volume contains logs organized in daily files, one for each Enghouse Reporting container.

Using Linux commands like `grep`, `less` and `tail`, you can read and analyze Enghouse Reporting logs.

Troubleshooting database connection issues

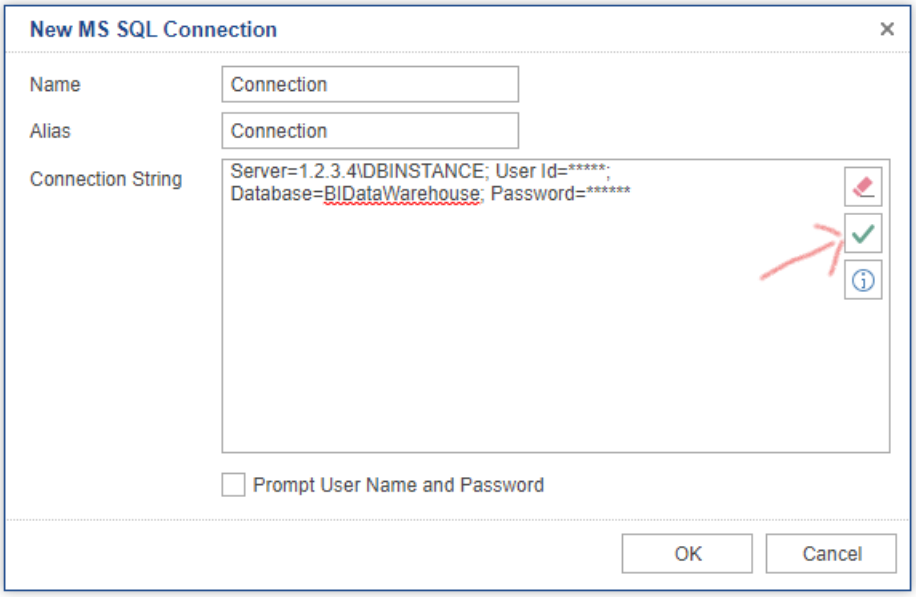
When a new tenant is configured, the connection string provided to the user interface is not tested yet. If you experience issues opening standard reports, usually, this means that a data connection check needs to be done.

To test a connection string, access Enghouse Reporting with a Data Designer user and create a new empty data model and then add a new data source:



Then, select the desired database type (e.g., MS SQL):

Finally, enter the connection string you want to test and click **Check**:



If the connection string is correct and the database is properly configured, you will receive a confirmation message. Otherwise, you will get an error message with information relevant to resolving the connection issue.

If you need to get the IP and/or the port of a SQL Server database, run the following query:

```
select distinct local_net_address, local_tcp_port
from sys.dm_exec_connections
where local_net_address is not null
```

Tenant management

> **Note**
> The following information refers to Reporting Version 5.X

Creating a New Tenant

> **Caution**
> The creation of a new tenant can require to create a new DWH specific for that tenant, and instal and configure a new instance of the ETL that loads new data into the DWH. DWH and ETL depend on the product of the data you are going to visualize with ERS (CCSP, CC, etc.) and your tenant isolation model (single or multitenant DWH).

1. Once you are logged in to Enghouse Reporting with the system.admin user, open the **Admin** tab, then click **Tenants** in the **Tenant management** section of the left-hand pane.
2. Click **+ NEW TENANT** to configure a new tenant.
3. In the **Tenant details** window, specify the following details:
    - **Tenant name**: The name you want to assign to the new tenant.
    - **Dashboarding enabled**: Select the checkbox to enable Dashboarding.
    - **Number of days to warn the user before license expires**: Select how many days prior to your license expiring you would like to be reminded that your license is about to expire.

      > **Note**
      > If you choose **0**, you will not be reminded of your license expiring.

    - **First day of the week**: Starting day of the week for calendar-related reports and visualizations. In the dropdown menu, you can choose between Sunday and Monday.
    - **Hosts**: Specify one or more hosts that users can use to access the new tenant; e.g., if the tenant uses the URL https://bireporting.example/ to access EnghouseReporting, the host for the tenant is `bireporting.example`.

      > **Caution**
      > Due to a bug in version 4.1.1 and previous versions, you need to add the host twice, as follows:
      >
      > `bireporting.example`
      >
      > `https://bireporting.example`

    - **Database connections**: Start filling out the **Connection string** box to make additional options appear.
        - **Name**: The name you want to assign to the database connection.

          > **Note**
          > If you want to use Standard Reports, you need to create a database connection called **DWH** (using capital letters).

- **Database type**: From the dropdown list, choose the type of the tenant's database. The currently available options are Oracle, Microsoft SQL Server, PostgreSQL, and MySQL.
- **Connection string**: The connection string to the standard DWH that contains call center data for the new tenant. It is the connection string that will be assigned to standard reports when the tenant users use them. The connecting string format is typically like the following: `Server=<serverIpOrName>; User Id=<dbUser>; Database=<databaseName>; Password=<userPassword>`. For help with verifying the connection string, see *Troubleshooting database connection issues on page 32*.

> **Note**
> To add a database connection, click the **Add connection** button. If you wish to add more than one connection, repeat the above-mentioned process. The connection string is checked when set for a tenant; if the connection is invalid, an error is shown to indicate what the issue is.

- **Tenant administrator**: The first admin user that is created for the new tenant. This user is the tenant admin that can be utilized to start using and configuring the tenant. Enter the following information:
  - **Email**: The email address for the admin user (e.g., admin@tnt1.com).
  - **Username**: The username (e.g., admin) of the admin user.
  - **First name**: First name of the admin user.
  - **Last name**: Last name of the admin user.
  - **User language**: From the dropdown menu, choose the default language for the tenant administrator.
  - **Password**: The initial password for the admin user.
  - **Confirm password**: The password must match the first entry.

4. Click **Save** to add the new tenant to your system.

When a new tenant is created, it is recommended to update the default translations to be applied in reports for the tenant. In order to apply standard translations in the tenant DWH, you have to access the `translations-importer-cli.service` container. This is done with the following command:

```
sudo docker-compose exec translations-importer-cli.service /bin/sh
```

Then run the following command, applying the relevant settings needed to connect to the tenant's DWH:

```
node build/translations-importer-cli.min.js -d <DWH type> -u <DWH user> -p '<DWH password>' -s <DWH host> -t <DWH port> -n <DWH name> -f ./translations
```

The translations folder is mapped from the installation folder of Enghouse Reporting. In order to manage custom translations, you can create subfolders in it and store the custom translations. For example, if you create a CCSP folder in `/home/enghouse/bireporting/translations`, you can then store the custom translations for Enghouse Reporting standard reports required for a specific tenant, or for more than one tenant, in that folder. In order to load custom translations from that folder, you need to run the following command within the `translations-importer-cli.service` container:

```
node build/translations-importer-cli.min.js -d <DWH type> -u <DWH user> -p '<DWH password>' -s <DWH host> -t <DWH port> -n <DWH name> -f ./translations/CCSP -c
```

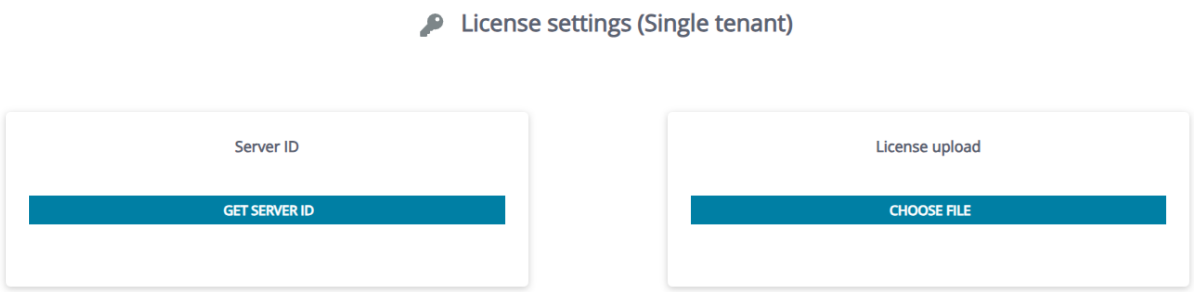
Note the `-c` option applied to the last command; it is needed to mark translations as custom, which stops the standard translations from overriding the custom ones.

This procedure can be used whenever a new version of the translations is provided.

It is recommended to always run the standard translations first, and the custom ones after.

Licensing a Tenant

When a new tenant is created, open https://<baseURL> to log in with the tenant Admin user, as defined during tenant creation, and then to license the tenant. At first login, the only option available to the user is the **License settings (Single tenant)** page.



The user has to get the server ID in order to request the license through the Enghouse BI Licensing Portal.

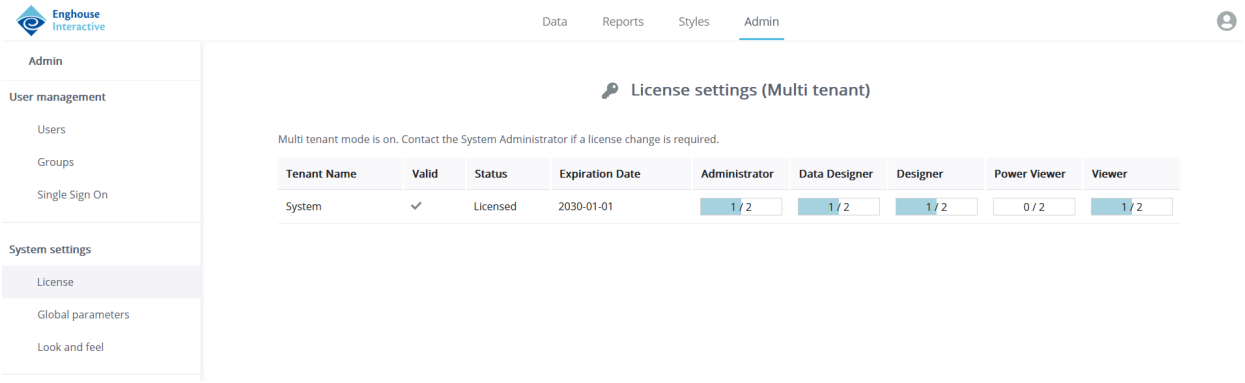
Once the license is provided, the tenant Admin needs to upload the license using the same UI. to do so, do the following:

1. Click **Choose file** to upload a valid license.
2. A window detailing the license information pops up. Check to see if the information is correct.
3. Click **Upload**.

After tenant licensing, start working with the Enghouse Reporting application.

Multi-tenant licenses

An Admin user with a multi-tenant license cannot see the server ID nor can they upload a license. The Admin only sees a message stating they should contact the System Administrator to set a valid license for their tenant. Ideally, the System Admin will have already allocated the licenses for the tenant after the creation of the license. If so, the Admin sees the valid license information.

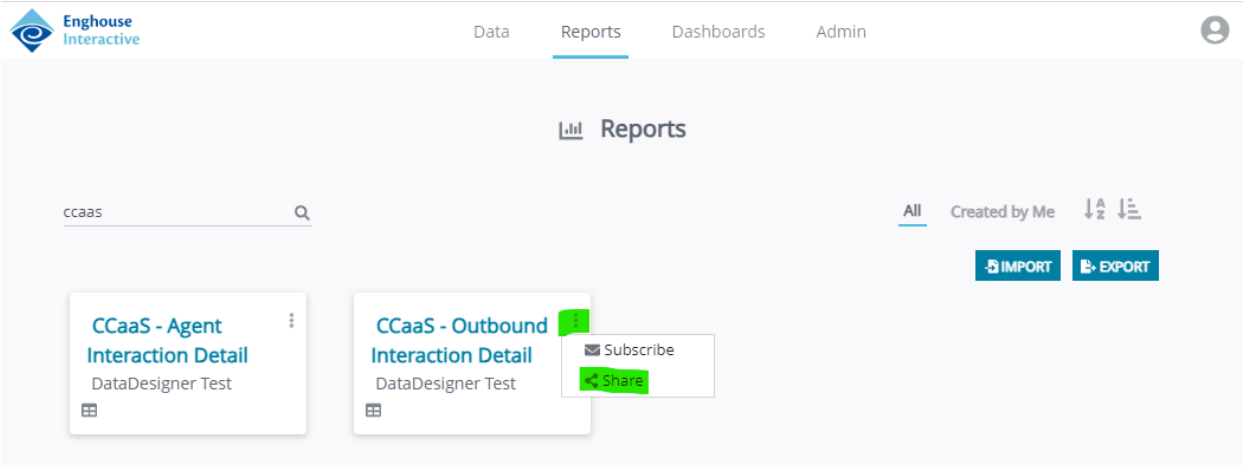


Sharing assets

When a new tenant is created, all the data models, base reports and report views are automatically published for the new tenant, but nothing is shared with users.

The tenant administrator must share the required data models and visualizations with the appropriate users.

To share a data model, a report or a view, login with an admin user, identify the asset you want to share in the data or reports page, click on the three dots options menu in top right corner of the asset tale, then click **Share**:



Alternatively, you may wish to share multiple assets at the same time. To do so, use the bulk operations view  , select and share assets in bulk.

Then, search for the user or group you want to share the asset with and click the name of the user or group. By default, it will be shared with view options only, but you can change it to allow users to edit the asset as well.

Repeat the last step for each user and group you want to share the asset with.

Finally, click **Save** to confirm the changes.

> **Caution**
>
> To allow users to properly access reports and views, it is required that the underlying data model is shared as well. Pay attention to properly share all the required data models when sharing reports and views.

Updating a Tenant

Once you are logged in to Enghouse Reporting with the system.admin user, open the **Admin** tab, then click **Tenants** in the **Tenant management** section of the left-hand pane. You will see a list of configured tenants with a subset of the current settings.

To update a tenant configuration, click **Edit** in the rightmost column of the table, in the row related to the tenant you want to update. The edit form is displayed, and you will be able to edit the following tenant settings:

- **Name**: The name assigned to the tenant.
- **Enabled**: Flag that allows you to enable or disable the tenant.
- **Dashboarding enabled**: Select the checkbox to enable Dashboarding.
- **Number of days to warn the user before license expires**: Select how many days prior to your license expiring you would like to be reminded that your license is about to expire.
- **First day of the week**: Starting day of the week for calendar-related reports and visualizations. In the drop-down menu, you can choose between Sunday and Monday.
- **Hosts**: Specify one or more hosts that users can use to access the tenant.
- **Database connections**: Start filling out the **Connection string** box to make additional options appear.
  - **Name**: The name you want to assign to the database connection.
  - **Connection string**: The connection string to the standard DWH that contains product data for the tenant. It is the connection string that will be assigned to standard reports when the tenant users use them. The connecting string format is the following: `Server=<serverIpOrName>; User Id=<dbUser>; Database=<databaseName>; Password=<userPassword>`
  - **Database type**: Choose one of the database types supported by EnghouseReporting.

Setting the Tenant SSO

Once you have logged into Enghouse Reporting with an admin user of a specific tenant, open the **Admin** tab and click **Single Sign On** in the **User management** section of the left-hand pane.

Here you can enable or disable the SSO and, when enabled, you can choose among the available SSO methods - JWT or OIDC.

If you select JTW, you will be prompted to enter the following attributes:

- **Remote Login URL**: The login page that allows the system to authenticate the user.
- **Remote Logout URL**: The logout page that allows the system to terminate a user session.
- **Shared Secret**: The JWT shared secret that the system uses to verify the JWT authentication.

If you select OIDC, will be prompted to enter the following attributes:

- **Issuer**: The issuer URL provided by your OIDC identity provider, e.g., `https://<oidcServerAddress>/auth/realms/tnt`.
- **Client ID**: The name of the client defined in your OIDC identity provider, e.g., bi.reporting.
- **Client Secret**: The secret key provided by the OIDC identity provider for the OIDC client.
- **Scope**: Defaults to “openid” when not specified.
- **Challenge Method**: Defaults to “S256” when not specified.

> **Note**
>
> Once SSO has been enabled, by default the user will be forwarded to the identity provider authentication page to login; in some cases, it may be useful to access the system by bypassing the SSO and using the Enghouse Reporting internal users (e.g., using the tenant admin). To do that, enable access to Enghouse Reporting by adding `/Account/Login?admin_only=true` to the base URL, e.g.:
>
> `https://reporting.sample.com/Account/Login?admin_only=true`

Deleting a tenant

Currently EnghouseReporting doesn't allow you to delete a tenant completely, but just disable it. When a tenant is disabled, tenant's users are no longer able to access the web application.

To disable a tenant, access EnghouseReporting with the system.admin user, open the **Admin** tab, then click **Tenants** in the **Tenant management** section of the left-hand pane. A table with tenants information will be displayed on your screen. There, you can identify the tenant that you have to disable from the entire list of tenants or use the search feature to easily find it. Once you find the tenant row, click the **Edit** action on the rightmost column of the table; from the tenant settings, disable the flag under the **Enabled** option and save the changes by clicking the **Save** button at the bottom-right corner.

If a tenant has to be deleted and there is no reason to think you will need to re-activate it, you should also consider cleaning up all the analysis data and freeing up the resources used for this tenant; the main elements that use resources for a tenant are the ETL and the DWH. Depending on the product that loads data into the DWH and the tenant isolation model you applied on the DWH (single or multitenant DWH), cleaning up resources could be more or less simple and fast.

> **Note**
> A disabled tenant can be enabled again in EnghouseReporting, but if you already uninstalled or deconfigured the ETL and dropped or cleaned up the DWH, enabling the tenant in ERS is not useful, as you have to configure DWH and ETL for managing the tenant's data again.

Single tenant DWH and ETL

When your integration uses an independent database and a dedicated instance of the ETL for the tenant, cleaning up resources is just matter of dropping the database or the database schema specific for the tenant, as well as deleting or uninstalling the ETL instance.

Multitenant DWH and ETL

When DWH and ETL are shared with other tenants, disabling the ETL is usually just a matter of reviewing the ETL configuration to disable the specific tenant, depending on the integration implementation.

Cleaning data from the DWH is currently not supported with standard tools as it requires deleting tenant-specific data from each DWH structure, an activity that can take many hours to complete.

A SQL script may be developed in future releases to clean up tenant data.