



Components

Forterro Deutschland Abas GmbH

Date: 17.03.2025

Table of Contents

1. Requirements	2
2. Basic installation of the components	3
2.1. Optional: Download and unpack components-v1-latest.tgz	3
2.2. ToDo: Unpack the components-<version>.tgz in the standard release of the Abas version	3
2.3. ToDo: Create or adapt the configuration file	3
2.4. ToDo: Create/customize .server.conf	6
2.5. Installation script	6
2.6. ToDo: Install components (as the s3 user)	6
3. Encryption via SSL certificates	11
3.1. Requirements	11
3.2. Use of certificate chains	12
3.3. Configuration of the certificates in the components	13
4. Upgrade of components	14
4.1. Upgrade of an existing component installation	14
4.2. Upgrade the Abas installer modules to the components	15
5. Advanced configuration	18
5.1. Grafana for visualizing the component logs	18
5.2. Set up a dashboard user with credentials in Keycloak	18
5.3. Setting up an LDAP(S) configuration (optionally with Kerberos authentication) in Keycloak	22
5.4. Installation on distributed systems (from COMPONENTS_VERSION 1.0.1)	27

Refers to component versions: 1.x.x

1. Requirements

- To download the components, there must be an internet connection available for the Abas server.
- You will need your Extranet credentials for the download.
- You will need a "Tenant ID" to install.
- Docker Compose standalone version 2.20 (and higher) and Docker Engine version 20.10 (and higher) are required for the installation.
- For the installation, a "docker login" to the artifact repository used must have been successful beforehand.
- From components.tgz version 1.0.1 an SSH key for the connection from the component host to the Abas host must be set up (even if this is the same host).
- Certificates must be available for the installation that must comply with some known [prerequisites](#).
- The use of IPv4 is a mandatory requirement for the components.

2. Basic installation of the components

2.1. Optional: Download and unpack components-v1-latest.tgz

Use your Extranet credentials for authentication.

Laden Sie das neueste Archiv herunter und entpacken Sie es.

```
cd $(dirname $HOMEDIR)

wget --user <user> --ask-password https://abasartifactory.jfrog.io/artifactory/abas.downloads-releases/erp/components/components-v1-latest.tgz

tar -xzf $(dirname $HOMEDIR)/components-v1-latest.tgz
mv components $(stat --printf='%Ucomponents' $HOMEDIR)
```

2.2. ToDo: Unpack the components-<version>.tgz in the standard release of the Abas version



By default, the components.tgz that matches the Abas version is located in the \$HOMEDIR/archives directory. The COMPONENTS_VERSION entry in common-default.env indicates the version.

Entpacken Sie das Archiv z.B. parallel zur Abas Installation, z.B. mit dem Benutzernamen als Präfix dem das \$HOMEDIR gehört.

```
cd $(dirname $HOMEDIR)
tar -xzf $HOMEDIR/archives/components-<version>.tgz
mv components $(stat --printf='%Ucomponents' $HOMEDIR)
```

Während der Installation wird ein SSH-Zugriff vom Ort des components Verzeichnis auf das abas-HOMEDIR benötigt.

```
|---<abas-HOMEDIR> (gehört hier s3)
|   |-- <Mandant>
|   |-- <Mandant>
|---<s3components>
|   |-- bin
|   |-- common-default.env
|   |-- <Komponente>
|   |-- <Komponente>
|   | ...
```

2.3. ToDo: Create or adapt the configuration file

In parallel to **common-default.env**, create the file **common-custom.env** with the following

environment variables.

Inhalt der common-custom.env

```
HOST_NAME=  
CUSTOMER_ROOT_CA_CERTIFICATE=  
CUSTOMER_ISSUED_CERTIFICATE=  
CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY=  
CUSTOMER_CA_BUNDLE_PEM=  
B64_TENANT=  
ABAS_PASSWORD=  
# ab components.tgz Version 1.0.1  
ABAS_SSH_USER=  
ABAS_SSH_HOST=  
ABAS_SSH_PORT=  
HOST_DOMAIN=
```

The variables listed are **mandatory variables** and must be set before starting.

HOST_NAME

Host name or IP address of the host - the certificates must be issued for these, i.e. the combination of HOST_NAME + HOST_DOMAIN must be entered in the certificate as the Subject Alternative Name.

CUSTOMER_ROOT_CA_CERTIFICATE

Full path of your root CA certificate or an intermediate certificate.

Important: The components require read access to the file! Some of the components will trust this certificate and therefore implicitly also the CUSTOMER_ISSUED_CERTIFICATE, as it is stored in their trust store.

CUSTOMER_ISSUED_CERTIFICATE

Full path to a certificate signed with your root CA certificate. Keycloak and nginx (web server) will authenticate themselves with this certificate. All other components must be able to validate this using their trust store (in which the CUSTOMER_ROOT_CA_CERTIFICATE is imported) or the CUSTOMER_CA_BUNDLE_PEM.

Important: The components require read access to the file!

CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY

Full path to the private key of your signed certificate.

Important: The components require read access to the file!

CUSTOMER_CA_BUNDLE_PEM

Full path to a file that contains a **complete** certificate chain of Base64 certificates. This chain must consist, from top to bottom, of the **signed certificate, 0 to n intermediate certificates and finally the root CA certificate**

So copy the contents of all relevant certificates into this file and read [the notes on CUSTOMER_CA_BUNDLE_PEM](#) again if you are unsure. **Important:** The components require read access to the file!

B64_TENANT

Your tenant ID.

ABAS_PASSWORD

A password of an admin user in Abas using which the JWT component can login.

ABAS_SSH_USER (from components.tgz version 1.0.1)

SSH user on the Abas host. Can alternatively be passed to the installation script as `--abas-ssh-user` parameter.

ABAS_SSH_HOST (from components.tgz version 1.0.1)

Hostname of the Abas host. Can alternatively be passed to the installation script as `--abas-ssh-host` parameter.

ABAS_SSH_PORT (from components.tgz version 1.0.1)

Port of the Abas host. Can alternatively be passed to the installation script as `--abas-ssh-port` parameter.

The listed variables are **optional variables** and can be set before starting.

HOST_DOMAIN

Optionally, the **HOST_DOMAIN** variable can also be defined as a suffix for the `HOST_NAME` variable. This means, for example, `HOST_NAME=system_a` and `HOST_DOMAIN=.my-domain` will use `system_a.my-domain` for all components when starting.

COMPOSE_PROFILES (from components.tgz version 1.0.1)

Docker Compose standalone standard environment variable through which you can specify profile names separated by a comma. The profiles for services are predefined in `docker-compose.yml`.

ABAS_HOMEDIR (as of components.tgz Version 1.0.1)

HOMEDIR on the Abas server. If the variable is not set, the attempt is made to determine the HOMEDIR when starting the component. If you have problems determining the HOMEDIR, you can also define the variable in `common-custom.env`, or transfer using the option `-h` when calling the

installer.



Do not set values in `common-default.env` as this file will be overwritten during an update. Instead, use `common-custom.env` to set your own values.

2.4. ToDo: Create/customize `.server.conf`

To access a client via REST API, you must create a `.server.conf` file either for each client or globally in `$HOMEDIR`. In addition to connection information for the `license_controller` and `jwt_auth_userinfo` components, this configuration file also contains your installation number.

Inhalt der `.server.conf`

```
[LicenseController]
url = http://<host_fqdn>:<license_controller_port>
installation = <installation_number>

[SSO]
server = http://<host_fqdn>:<jwt_port>
verify_peer = false
```



The placeholders `<host_fqdn>`, `<license_controller_port>`, `<installation_number>` and `<jwt_port>` must be replaced with your values. The default ports of the components can be found in the `common-default.env` file.

2.5. Installation script

What does the `components_installer.sh` script do?

- The script is located in the components bin directory (`components/bin`).
- The script checks the required environment variables for installing the components.
- The script calls `create_rest_api_config_files.sh` to create configuration files for Abas clients under `components/rest-api/abasconfig`. You decide whether and for which clients this should be done.

If the required environment variables are set, all components can be installed.

2.6. ToDo: Install components (as the `s3` user)

Um die Docker-Images der Komponenten herunterzuladen, führen Sie zunächst einen Docker-Login an der Abas Docker-Registry aus:

```
docker login abasartifactory.jfrog.io
```


To check whether your environment configuration is complete, you can run the following command in the components bin directory:

```
components_installer.sh --home-dir $HOMEDIR --check
```

From components.tgz Version 1.0.1

```
components_installer.sh [--abas-ssh-user <s3>] [--abas-ssh-host <s3Host>] [--abas-ssh-port <s3Port>] --check
```

The environment variables are also checked prior to the installation. Therefore, you can also use the following call: (you can also specify a project name for the component start and start optional services via profile names)

```
components_installer.sh --home-dir $HOMEDIR --up [--service <docker-service-name>] [--project <docker-project-name>]
```

From components.tgz Version 1.0.1

```
components_installer.sh [--abas-ssh-user <s3>] [--abas-ssh-host <s3Host>] [--abas-ssh-port <s3Port>] [--service <docker-service-name>] [--project <docker-project-name>] --up
```

You will be asked which clients you would like to create REST API configuration files for. If you want to automatically create the REST API configuration files for all clients, you can specify this using the **--all-clients** parameter.

The following components are started via components_installer.sh:

- keycloak-postgres
- keycloak
- webserver
- rest-api
- jwt-auth-userinfo
- user-administration
- dashboard-mongodb
- dashboard-api
- dashboard
- license-controller
- freitexteditor
- bpm (optional, can be started via COMPOSE_PROFILES (comma-separated list) in the common-

custom.env)



For BPM, define an EDP_CONFIG and an ADMIN_USER_EMAIL in addition to the COMPOSE_PROFILES=with-bpm in the common-custom.env. You can find a template in the components/bpm/service-custom.env.template file. The BPM backend creates a user with this mail address in the BPM Postgres database. The ADMIN_USER_EMAIL should therefore match a user in Keycloak who has the rights to handle workflows in Abas. If BPM was not started with a correctly configured ADMIN_USER_EMAIL, either delete the fresh BPM Postgres database and configure it appropriately in components/bpm/service-custom.env or customize the user in BPM at the following URL: http://<host_fqdn>:8088/camunda/app/welcome/default/#/login

Über den Parameter `--down` können alle Komponenten gestoppt werden. Soll nur eine einzelne Komponente gestoppt werden, kann der Name der Komponente über `--service` übergeben werden.

```
components_installer.sh --down [--service <docker-service-name>] [--project <docker-project-name>]
```

Eine positive Konfigurationsprüfung sieht in der Konsole zum Beispiel wie folgt aus:

```
Abas component-installer script started at Wed Nov 29 11:09:01 CET 2023
HOMEDIR: /example/dir/s3
COMPONENTS_DIR: /example/dir/s3/components
PROJECT_NAME: s3
Check files for all clients:
  Check file /example/dir/s3/components/rest-api/abasconfig/erp.connection-eks.cfg for client eks
  ... already exists
  Check file /example/dir/s3/components/rest-api/abasconfig/erp.connection-demo.cfg for client demo
  ... already exists
Confirm environment variable configuration ...
  CUSTOMER_ROOT_CA_CERTIFICATE (config) :ok
  CUSTOMER_ROOT_CA_CERTIFICATE (file) :ok
  CUSTOMER_ISSUED_CERTIFICATE (config) :ok
  CUSTOMER_ISSUED_CERTIFICATE (file) :ok
  CUSTOMER_CA_BUNDLE_PEM (config) :ok
  CUSTOMER_CA_BUNDLE_PEM (file) :ok
  CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY (config) :ok
  CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY (file) :ok
  KC_HOSTNAME (config) :ok
  KC_PORT (config) :ok
  WEBSERVER_HOSTNAME (config) :ok
  WEBSERVER_PORT_HTTPS (config) :ok
  JWT_ABAS_PASSWORD (config) :ok
  JWT_PORT (config) :ok
  JWT_AUTH_USERINFO_PERMISSIONS (config) :ok
  JWT_AUTH_USERINFO_WORKSPACES (config) :ok
  B64_TENANT (config) :ok
  LICENSE_CONTROLLER_PORT (config) :ok
Check .server.conf files
  Reading /example/dir/s3/eks/.server.conf
  LicenseController (.server.conf) & LICENSE_CONTROLLER_PORT (env) (port-match: ok)
```

SSO (.server.conf) & JWT_PORT (env) (port-match: ok)

Eine negative Konfigurationsprüfung sieht in der Konsole zum Beispiel wie folgt aus:

```
Abas component-installer script started at Wed Nov 29 10:27:24 CET 2023
HOMEDIR: /example/dir/s3
COMPONENTS_DIR: /example/dir/s3/components
PROJECT_NAME: s3
Check files for all clients:
  Check file /example/dir/s3/components/rest-api/abasconfig/erp.connection-eks.cfg for client eks
... already exists
  Check file /example/dir/s3/components/rest-api/abasconfig/erp.connection-demo.cfg for client demo
... not existing
Confirm environment variable configuration ...
  CUSTOMER_ROOT_CA_CERTIFICATE ... CUSTOMER_ROOT_CA_CERTIFICATE not set !!
  CUSTOMER_ROOT_CA_CERTIFICATE ... not a file or not readable !!
  CUSTOMER_ISSUED_CERTIFICATE ... CUSTOMER_ISSUED_CERTIFICATE not set !!
  CUSTOMER_ISSUED_CERTIFICATE ... not a file or not readable !!
  CUSTOMER_CA_BUNDLE_PEM ... CUSTOMER_CA_BUNDLE_PEM not set !!
  CUSTOMER_CA_BUNDLE_PEM ... not a file or not readable !!
  CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY ... CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY not set !!
  CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY ... not a file or not readable !!
  KC_HOSTNAME ... KC_HOSTNAME not set !!
  KC_PORT ... KC_PORT not set !!
  WEBSERVER_HOSTNAME (config) :ok
  WEBSERVER_PORT_HTTPS (config) :ok
  JWT_ABAS_PASSWORD ... JWT_ABAS_PASSWORD not set !!
  JWT_PORT ... JWT_PORT not set !!
  JWT_AUTH_USERINFO_PERMISSIONS ... JWT_AUTH_USERINFO_PERMISSIONS not set !!
  JWT_AUTH_USERINFO_WORKSPACES ... JWT_AUTH_USERINFO_WORKSPACES not set !!
  B64_TENANT ... B64_TENANT not set !!
  LICENSE_CONTROLLER_PORT ... LICENSE_CONTROLLER_PORT not set !!
Check .server.conf files
  Reading /example/dir/s3/eks/.server.conf
  LicenseController-URL Port in .server.conf does not match LICENSE_CONTROLLER_PORT in env
  SSO-URL Port in .server.conf does not match JWT_PORT in env
```



Errors that prevent installation are marked with "!!". Other *information* should only be understood as such and may, where appropriate, be intentional.

Sofern alles in Ordnung ist, werden Sie gebeten die Installation mit dieser Konfiguration zu bestätigen,

Is this ok? Proceed with installation? (y/n)

woraufhin die Komponenten gestartet werden.

```
Running 12/12
  □ Container s3-keycloak-postgres-1   Healthy
0.0s
  □ Container s3-license-controller-1  Running
0.0s
```

```
□ Container s3-freitexteditor-1      Running
0.0s
□ Container s3-webserver-1          Running
0.0s
□ Container s3-dashboard-mongodb-1  Running
0.0s
□ Container s3-dashboard-api-1      Running
0.0s
□ Container s3-dashboard-1          Running
0.0s
□ Container s3-keycloak-1           Healthy
0.3s
□ Container s3-rest-api-1           Healthy
0.6s
□ Container s3-user-administration-1 Started
0.7s
□ Container s3-jwt-auth-userinfo-1  Started
0.6s
```

3. Encryption via SSL certificates

One aim when implementing the components was to encrypt the communication. SSL certificates are used for this purpose.

This chapter describes what is required for this and what to look out for in the certificates.



Some components are programmed in Java and use the standard Java validation for SSL certificates. In some cases, this is stricter than validation in a web browser or from `openssl`, for example.

It is therefore important to ensure that the certificates are issued correctly, as otherwise connection errors may occur between the components internally and other systems.

3.1. Requirements

The following table describes which files are required and which requirements apply to them:

Certificate/Key	Requirements
Root or Intermediate CA certificate	<ul style="list-style-type: none"> X509v3 Basic Constraints extension must contain <code>CA:TRUE</code> (see https://www.rfc-editor.org/rfc/rfc5280#section-4.2.1.9) File must be readable for all users
Signed certificate	<ul style="list-style-type: none"> The CN or the SAN must contain the <code>HOST_NAME</code> and the <code>HOST_DOMAIN</code> from the components configuration File must be readable for all users
Key for the signed certificate	<ul style="list-style-type: none"> File must be readable for all users
Intermediate certificate(s) (optional)	<ul style="list-style-type: none"> If the intermediate certificates are to be used, a certificate bundle and/or a PEM file must be created
Certificate bundle (optional)	<ul style="list-style-type: none"> Should contain all certificates in the following order: <ul style="list-style-type: none"> Signed certificate If available all Intermediate certificates Intermediate CA certificate if available Root CA certificate File must be readable for all users
PEM file (optional)	<ul style="list-style-type: none"> Should contain all certificates in the following order: <ul style="list-style-type: none"> Signed certificate If available all Intermediate certificates Intermediate CA certificate if available Root CA certificate File must be readable for all users

3.1.1. FQDN as CN or SAN in the signed certificate

When establishing an SSL connection, the client checks whether the server name in the signed certificate matches the server name to which it wants to connect. The connection is only established if this is the case.

The `HOST_NAME` and the `HOST_DOMAIN` of the server are defined in the configuration of the components. The components communicate with each other via the Fully Qualified Domain Name (FQDN for short), i.e. the combination of host name and domain.

The signed certificate must therefore contain the FQDN. This can be done either via the **Common Name (CN)** or via the **Subject Alternative Names (SAN)**.

Sample content of the SAN property of a signed certificate:

```
*Auszug Komponenten Konfiguration*
HOST_NAME=customer
HOST_DOMAIN=.customer-domain

*Auszug openssl x509 -in <signiertes Zertifikat> --text*
X509v3 Subject Alternative Name:
    DNS:customer, DNS:customer.customer-domain
```

Entries in the SAN property of an SSL certificate can be defined as a comma-separated list `DNS:<hostname>`, `IP:<ip-address>` and definitions can start with `<email:>`, `<URI:>`, `<DNS:>`, `<RID:>`, `<IP:>`, `<dirName:>`, `<otherName:>`.



If you use SAN entries in your certificate, make sure that they are valid (see <https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.6>).

If the SAN extension contains the following entries, for example: `DNS:test.com,URI:test.com`, the entry `URI:test.com` is considered invalid because the schema (e.g., `https://`) is missing (according to RFC5280, a URI entry **must** begin with a schema).

In Java applications, this means that the entire SAN extension (i.e. **all** SAN entries) is considered invalid and the connection to other systems may fail.

3.2. Use of certificate chains

In order for the applications to be able to verify themselves across the entire certificate chain, all certificates must be combined in one file.

The order of the certificates in the certificate chain should be as follows:

1. Signed certificate
2. If available, all Intermediate certificates
3. If available Intermediate CA certificate
4. Root CA certificate

A certificate chain can have the file extensions `.crt` or `.pem`, for example.

In the following example, a certificate bundle and a PEM file are created:

```
# Erstellung eines Zertifikats-Bundles
cat <signiertes Zertifikat>.crt <intermediate Zertifikate>.crt <Root CA Zertifikat>.crt >> cert-
bundle.crt

# Erstellung einer PEM-Datei
cat <signiertes Zertifikat>.crt <intermediate Zertifikate>.crt <Root CA Zertifikat>.crt >> cert-
bundle.pem
```



In this example, the certificate bundle and the PEM file have the same content. However, the PEM file may also contain private keys, e.g., which are not typically contained in CRT files.

3.3. Configuration of the certificates in the components

The component configuration expects 4 certificate files [compare configuration parameters](#):

```
CUSTOMER_ROOT_CA_CERTIFICATE=<Pfad zum Root oder Intermediate CA Zertifikat>
CUSTOMER_ISSUED_CERTIFICATE=<Pfad zum signierten Zertifikat, zum Zertifikats-Bundle oder zur PEM-
Datei>
CUSTOMER_CA_BUNDLE_PEM=<Pfad zur PEM-Datei>
CUSTOMER_ISSUED_CERTIFICATE_PRIVATE_KEY=<Pfad zum Key des signierten Zertifikats>
```

The variable **CUSTOMER_ISSUED_CERTIFICATE** can contain only the signed certificate as well as a certificate chain.

4. Upgrade of components

4.1. Upgrade of an existing component installation

If you are still using the Abas Installer for ERP versions prior to Abas 2024.Q3, skip this subchapter and go to [Upgrade the Abas Installer modules to the components](#)



As of COMPONENTS_VERSION 1.2.5 it's possible to automatically update the components to the latest version.

This feature was deactivated with the release of COMPONENTS_VERSION 1.4.1 and no longer works.

Instead, you can now explicitly specify the version you wish to upgrade to.



If a COMPONENTS_VERSION 1.2.9 or lower is currently installed and you would like to upgrade to COMPONENTS_VERSION 1.4.0 or higher, for example, it is necessary to perform an intermediate upgrade to version 1.3.0.

Should you accidentally start the upgrade to version 1.4.0 before the interim upgrade, the upgrade script will display a corresponding message.

There are two ways to upgrade an existing component installation:

1. Automatic upgrade to a specific version directly from the current installation directory (as of COMPONENTS_VERSION **1.4.1**, see common-default.env)
2. Manual upgrade to any released version from the components directory of the newly downloaded version

In both cases, the script `<components-directory>/bin/components_upgrade.sh` is used.

4.1.1. Automatic upgrade to a specific version (as of COMPONENTS_VERSION 1.4.1)

To upgrade to a specific COMPONENTS_VERSION, you can call the following script directly in the component directory currently in use as of COMPONENTS_VERSION **1.4.1**:

```
bin/components_upgrade.sh --version <version>
```

Replace `<version>` with the version you want to use. For an overview of the available versions, please visit <https://abasartifactory.jfrog.io/artifactory/abas.downloads-releases/erp/components/> (login required). You only need to specify the version, e.g., `bin/components_upgrade.sh --version 1.4.2`.

In this case, the script downloads the `components-<version>.tgz`, considers the current directory as `--old-components-dir` and overwrites the standard release contents (no custom files).

4.1.2. Manual upgrade to any version

To upgrade between component versions to any released version, the following steps are necessary:

1. [Optional: Download the components-<version>.tgz](#)

2. **ToDo:** `Unpack the components-<version>.tgz`
3. Wechsel in das neu entpackte Komponenten-Verzeichnis und Aufruf des Upgradeskripts unter Angabe des alten Komponenten-Verzeichnisses:

```
bin/components_upgrade.sh --old-components-dir <zu-aktualisierendes-Verzeichnis>
```

In this case, the script overwrites the "old" directory (no custom files) with the contents from the already downloaded components-<version>.tgz.

4.2. Upgrade the Abas installer modules to the components

The steps listed in this subchapter are only necessary if:

1. you have an Abas version prior to Abas 2024.Q3.
2. you have installed the Abas installer modules so far.



Note: Switching to the components means that communication is SSL encrypted. The standard Rest API (e.g. connection to dashboards) can no longer be reached on the old HTTP port after the changeover, but only via the web server (https://<host_fqdn>:4443).

4.2.1. ToDo: Stop the Abas Installer modules

Before you install the components, first stop all Abas Installer modules. To do this, you can use, for example, the following command:

```
/usr/local/bin/abas-installer services --stop -m <modul_verzeichnis>
```

4.2.2. ToDo: Adopt the data of the Abas Installer modules

To transfer the data of the Abas Installer modules to the components, proceed as follows:

1. **Optional:** `Download the components-<version>.tgz`
2. **ToDo:** `Unpack the components-<version>.tgz`



The data from the Keycloak module cannot be adopted and must be created again in the new Keycloak component after installation.

Transfer of the REST API configuration files

Copy the REST API configuration files from `${BASE_MANDANTDIR}/modules/rest-api/abasconfig/` to `${BASE_MANDANTDIR}/components/rest-api/abasconfig/`.

Transfer of the dashboards

To transfer your dashboards from the Abas Installer modules to the components, the following

requirements must be met:

1. The old MongoDB container of the Abas Installer modules must be started.
2. The new MongoDB container of the components must be started. If this is not yet the case, continue with chapter [ToDo: Create or adapt the configuration file](#).
3. No dashboards have yet been created in the component installation.

Dump the database and copy it to the local system

First, it is necessary to export the database from the old MongoDB container of the Abas Installer modules:

```
# Dump der Datenbank erstellen und nach /tmp/dashboard speichern
docker exec -it abas_mongodb-abas_mongodb-1 mongodump -u <admin-user> -p <admin-password>
--authenticationDatabase admin -d dashboard -o /tmp/dashboard

# Datenbank-Dump auf lokales Dateisystem kopieren
docker cp abas_mongodb-abas_mongodb-1:/tmp/dashboard /tmp
```

Rename the collection



Always check whether the name of the exported collection contains the correct tenant. Renaming may also be necessary if the Base64 string has not changed.

In order for the data to be displayed correctly in the new dashboard, the collection may have to be renamed. The collection name is composed as follows: **dashboards.v3.<tenant>**.

To do this, first check your current collection name:

```
# Dateinamen prüfen
cd /tmp/dashboard/dashboard
ls -l
-rw-r--r-- 1 s3 abas 52516 Aug  6 08:09 dashboards.v3.on_premise.bson
-rw-r--r-- 1 s3 abas  191 Aug  6 08:09 dashboards.v3.on_premise.metadata.json
```

Inhalt der dashboards.v3.on_premise.metadata.json

```
{
  "indexes": [
    {
      "v": {
        "$numberInt": "2"
      },
      "key": {
        "_id": {
          "$numberInt": "1"
        }
      },
      "name": "_id_"
    }
  ]
}
```

```
] ,
  "uuid": "28a4527c9b494340ad1e53aa51fe0657",
  "collectionName": "dashboards.v3.on_premise",
  "type": "collection"
}
```

As can be seen here, the tenant name **on_premise** was used during the module installation (occurs in file names and in line 16 in metadata.json).

Depending on the installation, it is possible that the wrong tenant name was used in the Abas Installer modules. To find out your tenant name, you can decode the B64_TENANT. To do this, use the command `echo -n "<base64-string>" | base64 -d`. The output should look like this:

Dekodierter Inhalt des B64_TENANT

```
{
  "tenant": "test",
  "tenant_id": "*****_****_****_****_*****",
  "domain": "*****",
  "aws_access_key_id": "*****",
  "aws_secret_access_key": "*****",
  "aws_region": "eu-central-1",
  "stage": "****"
}
```

In this case, the tenant name would be **test** (line 2). You can now rename the collection using the following commands:

```
# Dateien umbenennen
mv dashboards.v3.on_premise.bson dashboards.v3.test.bson
mv dashboards.v3.on_premise.metadata.json dashboards.v3.test.metadata.json

# collectionName in metadata.json umbenennen
sed -i 's/dashboards.v3.on_premise/dashboards.v3.test/g' dashboards.v3.test.metadata.json
```

Transfer data to new MongoDB container

You can now copy the database dump into the new MongoDB container and import it into the database:

```
# Datenbank-Dump in neuen MongoDB-Container kopieren
docker cp /tmp/dashboard s3components-dashboard-mongodb-1:/tmp

# Datenbank-Dump in MongoDB importieren
docker exec -it s3components-dashboard-mongodb-1 mongorestore -u <admin-user> -p <admin-password>
--authenticationDatabase admin --drop /tmp/dashboard
```

If you now call "My dashboards" in the dashboard web interface, you should be able to see your old dashboards.

5. Advanced configuration

5.1. Grafana for visualizing the component logs

5.1.1. Installation

components.tgz in the /bin directory contains a script that enables the installation of Grafana, Promtail and Loki.

```
grafana.sh
```

Promtail transmits the components metrics to Loki (centralized logging and log analysis), which in turn are visualized by Grafana.

Call the script and follow the instructions.

First, a configuration is required, which you create using the following caller:

```
grafana.sh --config
```

You can then start the installation:

```
grafana.sh --start
```

The three containers can be stopped with the following command:

```
grafana.sh --stop
```

5.1.2. Setting up a dashboard in Grafana

To do this, call the URL https://<host_fqdn>:3000 and log in with the standard credentials (admin:admin). You will be prompted to set a new password.

You can add data sources under "Connections", "Data sources". You should see an existing data source here, of type Loki with URL <http://loki:3100>.

You can create a new dashboard under "Dashboards". Select "Add new panel" and "Query" to create a query. Select the data source "Loki" and enter a query, e.g., `{project="components"}`. You can change the type of panel for visualization, e.g., to "Logs".

Detailed instructions for setting up Grafana can be found in the official [documentation](#).

5.2. Set up a dashboard user with credentials in Keycloak

5.2.1. Requirement for functionality

Before installing the components, you may have already configured an SSO entry in a `.server.conf` file and set the `ABAS_PASSWORD` in `components/common-custom.env`. If not, you will find details about this in the [Adjust component configuration](#) section.

5.2.2. Setup

[Users](#) > `erp_dashboard_user`

Erp_dashboard_user

- Details
- Attributes
- Credentials
- Role Mappings
- Groups
- Consents
- Sessions





ID	47d42f5a-34d5-47d3-b426-173596808eee
Created At	11/29/23 1:50:59 PM
Username	erp_dashboard_user
Email	erp_dashboard_user@abas.de
First Name	
Last Name	
User Enabled 	<input checked="" type="checkbox"/> ON
Email Verified 	<input type="checkbox"/> OFF
Required User Actions 	Select an action...
Impersonate user 	<input type="button" value="Impersonate"/>
	<input type="button" value="Save"/> <input type="button" value="Cancel"/>

Figure 1. Let's set up a user in the Abas Realm in Keycloak with which we can view Abas data in the Dashboard.

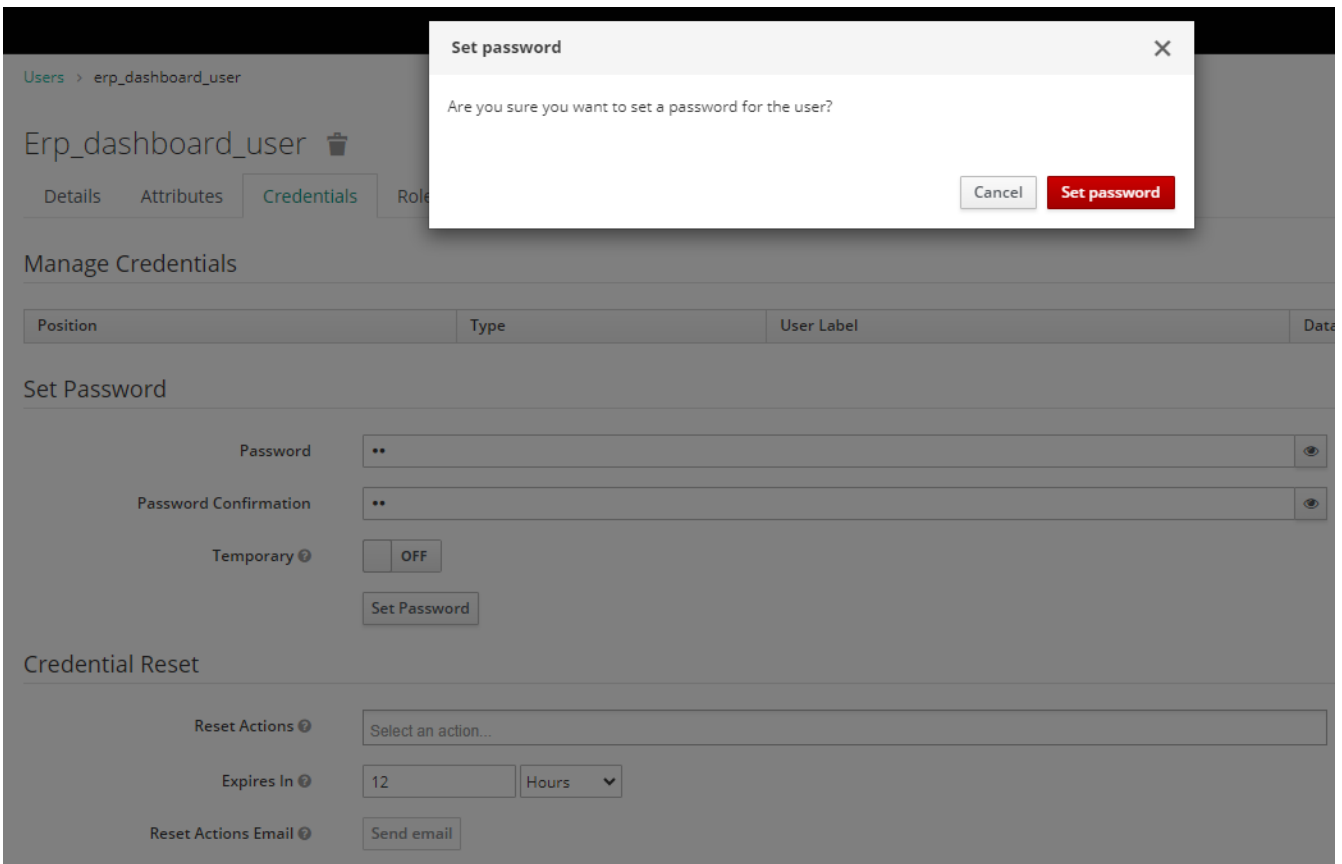


Figure 2. In the next step, assign the credentials (username and password) that you want to use when logging in to the Dashboard.



Pay particular attention to the email address! This must later match the email address of the user authorized for the desired use case in Abas.

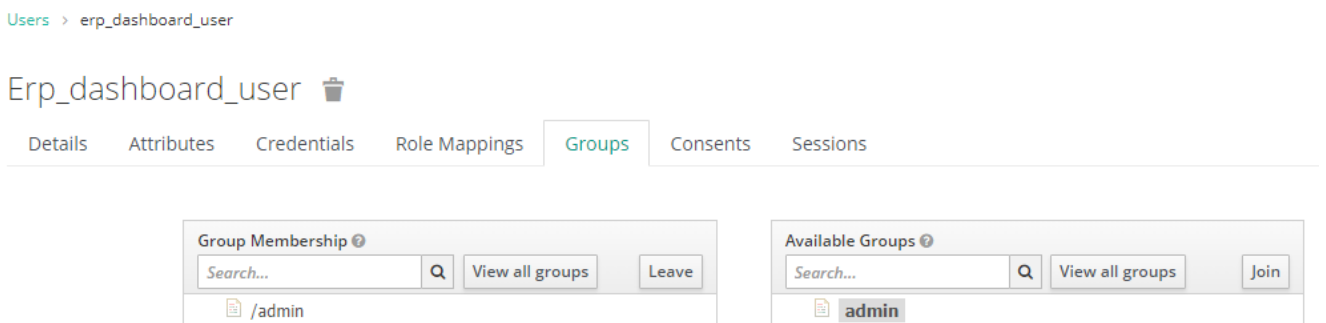


Figure 3. You then assign the user a corresponding group in Keycloak.

Passwortdefinitionen - zeigen [524 PASS erp_dashboard_user]

Datei Bearbeiten Ausführen Tabelle Kommando Fenster Info Hilfe

Neues Passwort Erlaubnisse zeigen

Allgemeines Kommandos GUI Zugangsrechte

Identnummer Suchwort

Bearbeiterzeichen Abas ERP Login

DMS-Benutzername Systemlogin

SSO-Login

Name Editieren

E-Mail

Externer Username

Abteilung Mitarbeiter

EINSTELLUNGEN

Hilfesprache Inaktiv

Bediensprache Fremd nutzbar

Druckeinstellungen

JFOP-Server-Konfiguration Matchcode-Erkennung

Objektauswahl anpassbar

Treffer der Objektauswahl dürfen kopiert werden

Figure 4. In Abas, log in as an admin user and set up a user in a password definition that matches the email address stored in Keycloak, who in this case has the necessary access rights for Dashboard.

5.2.3. Optional: addKeycloakUser.sh administration script

After the installation you can create the necessary users for Keycloak and expand the Abas "admin" user.



These adjustments function after a new installation. For an upgrade there probably isn't an "admin" user, or it has a changed password. The Keycloak users might also have to reference other users for an upgrade. In this case you'll need to use the script with additional options or create the user manually!

Usage vom Skript addKeycloakUser.sh

```
addKeycloakUser.sh -a -k
  -a: Add abas-ERP user (default: admin:admin with ID 27)
  -k: Add Keycloak user (default: admin:admin)

The default users can be overwritten with the following variables.
===== ATTENTION! =====
=== Only change these values if you know exactly what you are doing! ===
ERP_PASSWD_ID: ID of the abas-ERP password dataset (default: 27)
USERNAME:      abas-ERP username of the erp-Dataset
PASSWORD:      abas-ERP password of the erp-Dataset
EMAIL:         abas-ERP EMail of the erp-Dataset

KEYCLOAK_CREDENTIALS_ADMIN_USERNAME: Admin username (default: admin)
KEYCLOAK_CREDENTIALS_ADMIN_PASSWORD: Admin password (default: admin)
KEYCLOAK_PORT_HTTP: The keycloak port (default 8087)
```

Keycloak-Benutzer und abas ERP Benutzer anlegen

```
addKeycloakUser.sh -a -k
=== Add keycloak user
Generate token ...
Add user admin ...
Get user ID for admin ...
Get group ID ...
Add group to keycloak ...
Keycloak user successfully added!
=== Change abas ERP admin-user for dashboard
Update admin for client /mnt/abas/erp ... ok
```

5.3. Setting up an LDAP(S) configuration (optionally with Kerberos authentication) in Keycloak

You can also read about the topic in the official [documentation](#).

To set up an LDAP provider in Keycloak, the Keycloak supplied by Abas contains a configuration script that can be called manually from outside. The script for the call can be found in the directory of the components under `bin/call_ldap_user_provider.sh`.

5.3.1. Prerequisites: Existing certificates and Kerberos configuration files

For LDAPS:

You must be able to store the path to the root CA certificate or a corresponding certificate chain that matches your LDAP server in `components/common-custom.env`. The file must already exist.

For LDAP(S) with Kerberos authentication:

You must be able to store paths to both a keytab file and a `krb5.conf` file of your Kerberos server in `components/common-custom.env`. The files must already exist.

5.3.2. Optional: Storing an LDAP certificate for LDAPS

If you specify an `ldaps://` connection via the `CONNECTION_URL`, you may need to store an additional certificate or certificate chain that Keycloak should trust. To do this, add the following to `components/common-custom.env`:

`CUSTOMER_LDAP_CERTIFICATE`

Full path of your LDAP CA certificate.

Important: The component requires read access to the file! Keycloak will trust this certificate as it is stored in the internal trust store.

5.3.3. Optional: Configure LDAP(S) with Kerberos authentication

The following settings can be made for Kerberos in your `components/keycloak/service-custom.env`:

`KERBEROS_REALM`

Name of the Kerberos realm. For example: `FOO.ORG`

SERVER_PRINCIPAL

Full name of the server principal for the HTTP service including server and domain name. For example: HTTP/host.foo.org@FOO.ORG

5.3.4. Optional: Activate the Kerberos switches in the LDAP configuration script

If you want the LDAP configuration script to adopt settings for Kerberos, then you must add the keytab from Kerberos and the corresponding krb5.conf in your components/common-custom.env (you will find a krb5.conf.template file in the components/keycloak folder):

CUSTOMER_KERBEROS_KEYTAB

Complete path of your exported keytab of the Kerberos server. **Important:** The component requires read access to the file!

The keytab file is a security file used by Kerberos to facilitate the exchange of authentication information. It contains encrypted authentication keys for user principals or service principals. These keys allow a user or service to authenticate to a Kerberos system without the need to enter a password. The keytab file is typically stored on the user's or service's system and used by Kerberos services to authenticate users or services without requiring them to reveal their passwords.

CUSTOMER_KRB5_CONF

Complete path of your krb5.conf of the Kerberos server. **Important:** The component requires read access to the file!

The krb5.conf file defines the configuration of the Kerberos client on a Linux system. It contains information about Kerberos realms, KDCs (Key Distribution Centers) and administrative servers as well as encryption settings and other configuration options. The file enables the Kerberos client to perform proper authentication on the network and to interact with the Kerberos server.

5.3.5. ToDo: Add LDAP configuration for components

In the components/keycloak/service-default.env file you will find a section "LDAP configuration". The section is divided into two sections:

- 1. Variables to control the running of the LDAP configuration script
- 2. Variables for setting the values of the LDAP configuration in Keycloak



Please copy the contents of both sections into your components/keycloak/service-custom.env and configure them there, as no custom files will be overwritten during an update.

In the 1st section "KEYCLAOK ENTRYPOINT LDAP SCRIPT VALUES" you can make the following settings:

TEST_LDAP_CONNECTION

Should a connection test be performed using the LDAP connection data configured in section 2 before

the action is carried out? (Default is 'true')

TEST_LDAP_AUTHENTICATION

Should an authentication test be performed using the LDAP connection data configured in section 2 before the action is carried out? (Default is 'true')

LDAP_CONFIGURATION_ACTION

What action should be performed in the configuration script while starting Keycloak? Possible options are 'create' (does not change anything if it already exists), 'delete' (deletes a configuration found under the USER_PROVIDER_LDAP_ID and does not create a new one) and 'update' (calls delete and create in one step)

TRIGGER_FULL_SYNC

Should a full sync be triggered at the end of the LDAP configuration script if there is an LDAP configuration created at the time? (Default is 'true')

In the 2nd section "LDAP CONFIG VALUES" you can set, among other things, the following values of the LDAP configuration in Keycloak:

The first three values are required for the connection and authentication test before the configuration action.

CONNECTION_URL

Connection URL to your LDAP server. Usually in the following format: ldaps://LDAP_HOST:636

BIND_DN

DN of the LDAP administrator used by Keycloak to access the LDAP server.
Example: CN=Administrator,CN=User,DC=demo,DC=example,DC=com
Group example: OU=test,DC=test,DC=datical,DC=net

BIND_CREDENTIAL

LDAP administrator password

ENABLED

If you disable this user merging provider, it will not be included in queries and imported users will be disabled and read-only until the provider is re-enabled.
Either 'true' or 'false'

USER_PROVIDER_LDAP_ID

ID for the provider to be created/updated. For example: abas-ldap

VENDOR

The Vendor configuration parameter in Keycloak's LDAP component specifies the type of LDAP server used. Aside from the general "other" category, specific provider options may include:

'ad' for Microsoft Active Directory.

'edirectory' for Novell eDirectory.

'rhds' for Red Hat Directory Server.

'opendj' for OpenDJ.

'openldap' for OpenLDAP.

'fedora' for Fedora Directory Server.

'tivoli' for IBM Tivoli Directory Server.

'apacheds' for Apache Directory Server.

'openldap' for OpenLDAP.

These values may vary depending on the specific version of Keycloak and the supported LDAP provider. The exact list of supported providers and their respective values can be found in the official documentation or in the corresponding release notes.

EDIT_MODE

Configuring the edit mode on the LDAP configuration page defines the user's LDAP update permissions.

READ_ONLY

You cannot change the user name, email, first name, last name and other assigned attributes. Keycloak will display an error if a user attempts to update these fields. Password updates are not supported.

WRITABLE

You can change the user name, email, first name, last name and other assigned attributes and passwords and synchronize them automatically with the LDAP storage.

UNSYNCED

Keycloak saves changes to user name, email, first name, last name and passwords in the local Keycloak storage so that the administrator has to synchronize this data with LDAP again. In this mode, Keycloak deployments can update user metadata on read-only LDAP servers. This option also applies to the import of users from LDAP into the local Keycloak user database.

AUTH_TYPE

Type of authentication method used in the LDAP binding process. It is used in most requests sent to the LDAP server. Currently, only the mechanisms "none" (anonymous LDAP authentication) or "simple" (authentication with login information and password) are available.

USERS_DN

Complete DN of the LDAP tree in which your users are located. This DN is the superordinate DN of the LDAP user. For example, it could be "ou=users,dc=example,dc=com", assuming that your typical user has a DN like "uid='john',ou=users,dc=example,dc=com".

USERNAME_LDAP_ATTRIBUTE

Name of the LDAP attribute that is mapped as the keycloak user name. For many LDAP server providers, this can be 'uid'. For Active Directory it can be 'sAMAccountName' or 'cn'. The attribute should be filled in for all LDAP user records that you want to import from LDAP to Keycloak.

RDN_LDAP_ATTRIBUTE

Name of the LDAP attribute that is used as the RDN (top attribute) of the typical user DN. Normally it is the same as the LDAP attribute user name, but it is not required. For example, it is common for Active Directory to use "cn" as the RDN attribute when the username attribute might be 'sAMAccountName'.

UUID_LDAP_ATTRIBUTE

Name of the LDAP attribute that is used as a unique object identifier (UUID) for objects in LDAP. For many LDAP server providers this is "entryUUID", but for some it is different. For Active Directory, for example, it should be "objectGUID". If your LDAP server does not support the concept of UUID, you can use any other attribute that should be unique for LDAP users in the tree. For example 'uid' or 'entryDN'.

USER_OBJECT_CLASSES

All values of the LDAP objectClass attribute for users in LDAP, separated by commas. For example: 'inetOrgPerson, organizationalPerson'. Newly created Keycloak users are written to LDAP with all these object classes and existing LDAP user entries are only found if they contain all these object classes.

CUSTOM_USER_SEARCH_FILTER

Used to filter the complete list of users and groups in the "User DN" node to the users and groups you want to import into Keycloak. Leave this field empty if you do not require an additional filter. Can use a filter like (mail=*) to include only users with an email address (excludes users with a work account). You can filter by groups or other criteria

SEARCH_SCOPE

If the node listed under "User DN" contains nested nodes with users, select "subtree". Otherwise, select "one level".
Set SEARCH_SCOPE=2 for "subtree" and SEARCH_SCOPE=1 for "one level".

FULL_SYNC_PERIOD

Period for full synchronization in seconds. Default=604800

CHANGED_SYNC_PERIOD

Period for the synchronization of changed or newly created LDAP users in seconds. Default=86400

BATCH_SIZE_FOR_SYNC

Number of LDAP users imported from LDAP to Keycloak in one transaction. Default=200

EXTRA_LDAP_CONFIG_ARGS

Additional arguments can be added in the form:

```
-s 'config.<configName>=["<value>"]'
```

For example, debug options can be activated with:

```
"-s 'config.syncRegistrations=["false"]' -s 'config.debug=["true"]"
```

5.3.6. ToDo: Caller of call_ldap_user_provider.sh

If Keycloak is already running, you can call the script `bin/call_ldap_user_provider.sh` in your `keycloak/service-custom.env` after the Ldap configuration has been completed, which starts the actual Ldap configuration script in the Keycloak Docker container with the latest configuration.

5.4. Installation on distributed systems (from COMPONENTS_VERSION 1.0.1)

If the components are to be installed on a server other than the Abas server, this can be done in only a few steps. The same requirements apply as described in the [Requirements](#) chapter.

5.4.1. ToDo: Download the components-<version>.tgz archive

Download the `components-<version>.tgz` archive as described in the [Optional: Download the components-<version>.tgz archive](#) chapter.

5.4.2. ToDo: Extract the archive

Change to the directory in which you want to extract the components. Then you can use the following command to extract the archive:

```
tar -xzf components-<version>.tgz
```

5.4.3. ToDo: Adapt the configuration file

To adapt the configuration file, proceed as described in the [ToDo: Create or adapt the configuration file](#) chapter.



It is important to record the SSH connection data for the s3 user of the Abas server in the variables `ABAS_SSH_USER`, `ABAS_SSH_HOST`, and `ABAS_SSH_PORT`. In addition, the component user must be able to authenticate themselves to the Abas server as the s3

user via an SSH key.

5.4.4. ToDo: Copy the Abas font types

The free text editor requires the Abas standard font types as well as the customer-specific font types. During a component installation, an attempt is made to automatically load the fonts from the Homedir and store them in the components directory under `freitexteditor/fonts`.

In the `docker-compose.yml` in the components directory, you can see how the fonts of the component are made available (in the 'free text editor' under 'volumes').

Alternatively, the font types can also be copied manually. To do this, the `abas-fonts.jar` must be copied from the Abas server from `$HOMEDIR/java/lib/abas-fonts.jar` to `~/components/freitexteditor/fonts/abas-fonts.jar` on the component server. All custom fonts from the Abas server must also be copied from `$HOMEDIR/print/fonts` to `~/components/freitexteditor/fonts` on the component server.

5.4.5. ToDo: Set the host FQDN correctly in `.server.conf`

As described in the [ToDo: Create/Customize `.server.conf`](#) chapter, the license server and the SSO URL must be set in `.server.conf`. In the case of a distributed installation, ensure that the FQDN of the component server is entered in `.server.conf`.

5.4.6. ToDo: Continue with the installation

The installation can now be continued from the [ToDo: Install components \(as the s3 user\)](#) chapter.



When calling `components_installer.sh`, the message "WARN[0000] The "HOMEDIR" variable is not set. Defaulting to a blank string." will be displayed as no HOMEDIR variable is set on the component server. This message can be ignored. If you want to avoid the message, you can also create the HOMEDIR variable with blank string in `common-custom.env`.