univention
be open.

Keycloak app 21.1.2

Release 21.1.2

Univention GmbH

Aug 18, 2023

CONTENTS

Welcome to the documentation about the Univention **Keycloak** app. The app installs Keycloak[1], an open source software product for single sign-on with identity and access management. Furthermore, the app adds authentication to applications and secure services.

This documentation is for system administrators who operate the **Keycloak** app from Univention App Center connected to the LDAP directory in Univention Corporate Server (UCS). It covers the following topics:

1. *Installation* (page 3)
2. *Configuration* (page 9)
3. *Architecture* (page 29)
4. *Requirements and limitations* (page 31)
5. *Use cases* (page 33)
6. *Troubleshooting* (page 39)

This documentation doesn't cover the following topics:

- Usage of Keycloak itself, see the *Keycloak 21.1 Documentation* [1].
- Usage of UCS (Univention Corporate Server), see *UCS 5.0 Manual* [2].

To understand this documentation, you need to know the following concepts and tasks:

- Use and navigate in a remote shell on Debian GNU/Linux derivative Linux distributions like UCS. For more information, see Shell and Basic Commands[2] from *The Debian Administrator's Handbook*, Hertzog and Mas [3].
- Manage an app through Univention App Center[3] in *UCS 5.0 Manual* [2].
- Know the concepts of SAML (Security Assertion Markup Language[4]) and OIDC (OpenID Connect[5]) and the differences between the two standards.

Your feedback is welcome and highly appreciated. If you have comments, suggestions, or criticism, please send your feedback[6] for document improvement.

[1] https://www.keycloak.org/
[2] https://www.debian.org/doc/manuals/debian-handbook/short-remedial-course.en.html#sect.shell-and-basic-commands
[3] https://docs.software-univention.de/manual/5.0/en/software/further-software.html#computers-softwareselection
[4] https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language
[5] https://en.wikipedia.org/wiki/OpenID#OpenID_Connect_(OIDC)
[6] https://www.univention.com/feedback/?keycloak-app=generic

CHAPTER

ONE

INSTALLATION

You can install the **Keycloak** app like any other app with Univention App Center. The App Center only allows to install Keycloak on a UCS system with system role *Primary Directory Node* or *Backup Directory Node*. For more information, see Primary Directory Node[7] in *UCS 5.0 Manual* [2].

UCS offers two different ways for app installation:

- With the web browser in the UCS management system
- With the command-line

For general information about Univention App Center and how to use it for software installation, see Univention App Center[8] in *UCS 5.0 Manual* [2].

1.1 Installation with the web browser

To install Keycloak from the UCS management system, use the following steps:

1. Use a web browser and sign in to the UCS management system.
2. Open the *App Center*.
3. Select or search for *Keycloak* and open the app with a click.
4. To install Keycloak, click *Install*.
5. Leave the *App settings* in their defaults or adjust them to your preferences. For a reference, see *Settings* (page 20).
6. To start the installation, click *Start Installation*.

Note: To install apps, the user account you choose for login to the UCS management system must have domain administration rights, for example the username `Administrator`. User accounts with domain administration rights belong to the user group `Domain Admins`.

For more information, see Delegated administration for UMC modules[9] in *UCS 5.0 Manual* [2].

[7] https://docs.software-univention.de/manual/5.0/en/domain-ldap/system-roles.html#domain-ldap-primary-directory-node
[8] https://docs.software-univention.de/manual/5.0/en/software/app-center.html#software-appcenter
[9] https://docs.software-univention.de/manual/5.0/en/central-management-umc/delegated-administration.html#delegated-administration

1.2 Installation with command-line

To install the **Keycloak** app from the command-line, use the following steps:

1. Sign in to a terminal or remote shell with a username with administration rights, for example `root`.
2. Choose between default and custom settings and run the appropriate installation command.

   For installation with default settings, run:

```
$ univention-app install keycloak
```

   To pass customized settings to the app during installation, run the following command:

```
$ univention-app install --set $SETTING_KEY=$SETTING_VALUE keycloak
```

> **Caution:** Some settings don't allow changes after installation. To overwrite their default values, set them before the installation. For a reference, see *Settings* (page 20).

1.3 Initial Keycloak configuration

The first installation of the **Keycloak** app in the UCS domain creates an administrative user named *admin*, whose password is written to */etc/keycloak.secret*. With this user, the initial configuration of *Keycloak* is created.

To not overwrite custom settings, subsequent installations of the **Keycloak** app on additional UCS systems in the domain don't create the initial configuration.

Factory reset of the configuration

For a *factory reset* of the configuration, delete the realm *UCS* in the *Keycloak Admin Console*.

> **Warning:** Beware, your installation looses all custom configuration settings, custom services providers and other changes to the realm *UCS*.

After you deleted the realm *UCS*, create the initial configuration with the following command:

```
$ univention-keycloak --binduser=admin --bindpwdfile=/etc/keycloak.secret init
```

> **Warning:** After you deleted the realm *UCS*, create the initial configuration with the following command:

1.4 Sign in to Keycloak Admin Console

After a successful installation, signed in domain administrator users see the tile *Keycloak* on the UCS Portal, that directs them to the *Keycloak Admin Console*.

The URL has the following scheme: `https://ucs-sso-ng.$domainname/admin/`. The *$domainname* is your UCS domain name.

Example:

https://ucs-sso-ng.example.com/admin/

The username for login is the *name of the initial admin user* defined during installation and saved in the UCR variable `keycloak/admin/user`.

Note: All users in the `Domain Admins`, for example the domain user `Administrator`, can also sign in to the Keycloak Admin Console.

1.5 Fetch metadata for service provider configuration

OIDC (OpenID Connect) and SAML (Security Assertion Markup Language) both offer machine readable information to the services that want to use the authentication services in Keycloak. This information is the metadata discovery documents.

In the Keycloak Admin Console you can find them at *realm settings* ‣ *UCS* ‣ *Endpoints*. At the endpoints you see *OpenID Endpoint Configuration* and *SAML 2.0 Identity Provider Metadata*. To view the metadata discovery documents, click the endpoint entries.

With the following commands you can obtain the URLs to the metadata information. Some services comfortably take the URL and configure the authentication automatically.

To download the metadata information for OIDC, run the following command:

```
$ wget "https://ucs-sso-ng.$(hostname -d)/keycloak/realms/ucs/.well-known/openid-
↪configuration"
```

To download the metadata information for SAML, run the following command:

```
$ wget "https://ucs-sso-ng.$(hostname -d)/keycloak/realms/ucs/protocol/saml/
↪descriptor"
```

CHAPTER

TWO

UPDATE

You can update the **Keycloak** app like any other app with Univention App Center or the command line tool **univention-app**.

This chapter covers additional points to consider before and during the update process.

2.1 Configuration migration during Keycloak app updates

Some updates of the **Keycloak** app require a migration of the domain wide Keycloak configuration, because every Keycloak service in the UCS domain uses the same, shared configuration.

By default, the **Keycloak** app migrates the configuration during the app update process. Some changes may conflict with older versions of the **Keycloak** app. Therefore, make sure that all **Keycloak** app instances on all UCS systems in the domain have the same version.

Tip: You can deactivate the automatic configuration migration during the app update process. Set the value of the UCR variable `keycloak/auto-migration` to `false`.

You can manually migrate the configuration of the **Keycloak** app on all UCS system, **after** you updated the app on all UCS systems to the same version. Use the following command:

```
$ univention-keycloak upgrade-config
```

CHAPTER

THREE

CONFIGURATION

The **Keycloak** app offers various configuration options. Some settings don't allow changes after installation. Therefore, you must set them carefully **before** installation. You find those settings marked with *Only before installation* in *Settings* (page 20). You can change all other settings at any time after the installation.

To change settings after installation, sign in to the UCS management system with a username with administration rights and go to *App Center* ‣ *Keycloak* ‣ *Manage Installation* ‣ *App Settings*. On the appearing *Configure Keycloak* page, you can change the settings and apply them to the app with a click on *Apply Changes*.

The App Center then *reinitializes* the Docker container for the Keycloak app. *Reinitilize* means the App Center throws away the running Keycloak Docker container and creates a fresh Keycloak Docker container with the just changed settings.

3.1 Use Keycloak for login to UCS Portal

The **Keycloak** app can take over the role of the *SAML IDP* for the UCS Portal. And the portal can use Keycloak for user authentication.

Warning: The LDAP server will not recognize SAML tickets that the *simpleSAMLphp* based identity provider issued after you restart it. Users will experience invalidation of their existing sessions.

For more information about production use, see *Installation on UCS* (page 31).

To configure the UCS portal to use Keycloak for authentication, run the following steps on the system where you installed Keycloak:

1. Set the UCR variable `umc/saml/idp-server` to the URL `https://ucs-sso-ng.$domainname/realms/ucs/protocol/saml/descriptor`, for example `https://ucs-sso-ng.example.org/realms/ucs/protocol/saml/descriptor`. This step tells the portal to use Keycloak as IDP.

   Sign in to the UCS management system and then go to *System* ‣ *Univention Configuration Registry* and search for the variable `umc/saml/idp-server` and set the value as described before.

   Open a shell on the UCS system as superuser `root` where you installed Keycloak and run the following command:

   ```
   $ ucr set \
   umc/saml/idp-server=\
   "https://ucs-sso-ng.$(hostname -d)/realms/ucs/protocol/saml/descriptor"
   ```

2. Modify the portal to use SAML for login:

   In the UCS management system go to *Domain* ‣ *Portal* ‣ *login-saml*. On the tab *General* in the section *Advanced* activate the *Activated* checkbox.

   Open a shell on the UCS system as superuser `root` where you installed Keycloak and run the following command:

```
$ udm portals/entry modify \
--dn "cn=login-saml,cn=entry,cn=portals,cn=univention,$(ucr get ldap/base)" \
--set activated=TRUE
```

3. To activate the changes, restart the LDAP server `slapd` within a maintenance window.

   In the UCS management system go to *System ‣ System Services*. Search for `slapd` and click to select the service. Then click *Restart*.

   Open a shell on the UCS system as superuser `root` where you installed Keycloak and run the following command:

```
$ service slapd restart
```

Note: If you don't restart the LDAP server, you will see the following message in `/var/log/syslog`:

`slapd[…]: SASL [conn=…] Failure: SAML assertion issuer https://ucs-sso-ng.$domainname/realms/ucs is unknown`

By default **Keycloak** app creates a *SAML SP* (client) for every UCS Portal server. You can see the list of existing *SAML SP* clients with the following command:

```
$ univention-keycloak saml/sp get --json
[
    "https://ucs1.example.com/univention/saml/metadata",
    "https://ucs2.example.com/univention/saml/metadata",
    ...
]
```

If the *SAML SP* for a particular UCS Portal server doesn't exist, you can create it in **Keycloak** with the command:

```
$ FQDN="the fqdn of the UCS Portal server"
$ univention-keycloak saml/sp create \
  --metadata-url="https://$FQDN/univention/saml/metadata" \
  --umc-uid-mapper
```

3.2 Import of user attributes from UCS to Keycloak

Keycloak uses the LDAP directory of the UCS domain as backend for the user accounts. During the authentication process certain user attributes are imported into **Keycloak**. These attributes can be used later on in so called *Attribute Mappers* to pass additional information trough the SAML assertion or OIDC token to services (e.g. *displayName*).

By default the **Keycloak** app is configured to import the following user attributes:

LDAP attribute	Keycloak attribute
uid	username
uid	uid
entryUUID	entryUUID
lastname	lastName
mailPrimaryAddress	email
givenName	firstName
createTimestamp	createTimestamp
modifyTimestamp	modifyTimestamp

It is possible to configure the import of additional LDAP user attributes to **Keycloak**, for example

```
$ univention-keycloak user-attribute-ldap-mapper create description
```

to import the LDAP user attribute `description` to the **Keycloak** attribute `description`.

With the following command you get a list of all the currently configured **Keycloak** user attributes.

```
$ univention-keycloak user-attribute-ldap-mapper get --user-attributes
```

3.3 Keycloak as OpenID Connect provider

The **Keycloak** app can serve as an OpenID Connect provider (*OIDC Provider*). The following steps explain how to configure an OIDC relying party (*OIDC RP*) to use Keycloak for authentication:

1. *Keycloak Admin Console* (page 4).
2. Navigate to *UCS realm* ▸ *Clients* ▸ *Create*.
3. Specify the `client-id` for the client application (*OIDC RP*). Use the same `client-id` in the configuration of the client application.
4. Select `openid-connect` in the *Client Protocol* drop-down list.
5. Enter the *root URL*, the endpoint URL of the client application (*OIDC RP*).
6. Click *Save*.
7. Finally, the administrator can review the URL settings and customize them, if necessary.

For more information, see *Keycloak Server Administration Guide: OIDC clients* [4].

New in version 19.0.1-ucs1: **univention-keycloak** added. For more information about the usage, see the `--help` option.

As an alternative the app **Keycloak** offers a command line tool. For usage, see the following example:

```
$ univention-keycloak oidc/op/cert get \
--as-pem \
--output "$SOMEFILENAME"
$ univention-keycloak oidc/rp create \
--app-url="https://$(hostname -f)/${MYAPP_URL}/" "${MYAPP_CLIENT_ID}"
```

The option group `oidc/rp` offers additional options like `--client-secret`.

Note: If the administrator chooses `Confidential` as *Access Type* on the client configuration page, Keycloak offers an additional *Credentials* tab with the credentials.

3.4 Keycloak as SAML Identity Provider

New in version 19.0.1-ucs1: **univention-keycloak** added. For more information about the usage, see the `--help` option.

The **Keycloak** app can serve as an *SAML IDP*.

For apps that want to act as a *SAML SP*, you need to add a `client` configuration in Keycloak through the *Keycloak Admin Console* (page 4). For more information about how to create a SAML client configuration, see *Keycloak Server Administration Guide: Creating a SAML client* [5].

As an alternative the app **Keycloak** offers a command line tool. For usage, see the following example:

```
$ univention-keycloak saml/idp/cert get \
--as-pem --output "$SOMEFILENAME"
$ univention-keycloak saml/sp create \
--metadata-url "https://$(hostname -f)/$METADATA-URL-OF-THE-APP"
```

The option group `saml/sp` offers additional options like `--client-signature-required`.

Note: If the administrator chooses `Confidential` as *Access Type* on the client configuration page, Keycloak offers an additional *Credentials* tab with the credentials.

3.5 Backup and restore

Administrators can create a backup of the **Keycloak** app data. The data comprises information for example about the realm, clients, groups, and roles. To create a backup, run the *export* action as in the following steps:

```
$ univention-app shell keycloak /opt/keycloak/bin/kc.sh export \
--dir /var/lib/univention-appcenter/apps/keycloak/data/myexport
```

In this example `myexport` is a freely chosen directory name.

To restore the backup into the app **Keycloak**, run the *import* action as in the following step:

```
$ univention-app shell keycloak /opt/keycloak/bin/kc.sh import \
--dir /var/lib/univention-appcenter/apps/keycloak/data/myexport
```

Warning: **Keycloak** defines the scope of exported data and may not contain every configuration option the program offers.

3.6 MariaDB as database

The **Keycloak** app uses PostgreSQL as default database back end. This section explains how to configure the app **Keycloak** to connect and use a MariaDB database back end. The setup requires a configuration through *Settings* (page 20). Administrators can select the database back end either during initial app installation of **Keycloak** or change it later after installation.

The following examples for the database configuration assume that a user account with the appropriate permissions for MariaDB exists. They use the database user account `keycloak` and the password `database-password`.

Note: The database user needs the following minimum privileges to work in a single machine setup. Use the GRANT command[10]:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, REFERENCES, INDEX, DROP
ON `<database>`.* TO `<user>`@`<host>`;
```

To specify a MariaDB database during installation, run

```
$ univention-app install \
--set kc/db/url="jdbc:mariadb://${database_hostname}:3306/keycloak" \
--set kc/db/password="database-password"
```

[10] https://mariadb.com/kb/en/grant/

To specify a MariaDB database after installation in UMC:

1. Sign in to the UCS management system.
2. Go to *App Center* ‣ *Keycloak* ‣ *Manage Installation* ‣ *App Settings*.
3. Search for the variable `Database URI`. Set the value to your MariaDB endpoint, for example `jdbc:mariadb://$database_hostname:3306/keycloak` and click *Apply Changes*.

To specify a MariaDB database after installation on the command line:

```
$ univention-app configure keycloak \
--set kc/db/url "jdbc:mariadb://${database_hostname}:3306/keycloak" \
--set kc/db/password "database-password"
```

And to persist this change also in LDAP, use the following commands:

```
$ univention-install jq
$ new_json=$(univention-ldapsearch -LLL \
'(&(cn=keycloak)(univentionObjectType=settings/data))' \
| sed -n 's/^univentionData:: //p' | base64 -d | bzip2 --decompress \
| jq '.uri = "jdbc:mariadb://${database_hostname}:3306/keycloak"')
$ udm settings/data modify \
--dn "cn=keycloak,cn=data,cn=univention,$(ucr get ldap/base)" \
--set data=$(echo "$new_json" | bzip2 -c | base64 -w0)
```

3.7 Multiple installations in the domain

Administrators can install the app **Keycloak** on several nodes in a UCS domain to increase availability and provide failover using the default DNS name `ucs-sso-ng.$(hostname -d)`. The default installations in the domain don't require any interaction from the administrator. This will also provide session synchronization between all **Keycloak** installations on the domain.

Note: If the **Keycloak** app is installed on multiple systems in the domain and updates are available, make sure to update the app on all systems so that all instances of the app in the domain are on the same version.

3.8 Two-factor authentication for Keycloak

Warning: The two-factor capability is a built-in **Keycloak** feature that is not integrated into the UCS identity management or user lifecycle. More sophisticated integration needs to be added individually.

New in version 19.0.1-ucs1:

- Added functionality to enable 2FA (Two-Factor Authentication) to **univention-keycloak**. For more information about the usage, see the `--help` option.

The app **Keycloak** offers a 2FA option. 2FA is an authentication method that grants users access to a service after they sign in with a password and a OTP (one-time password) randomly generated by a third-party OTP password generator like *FreeOTP* or *Google Authenticator*.

2FA increases the protection for user data, because users need to provide two pieces: knowledge (password) and something in the users' possession (the OTP). It also increase the security of the system by avoiding account locking on known accounts because of malicious attacks. For more information, see Wikipedia: Multi-factor authentication[11].

[11] https://en.wikipedia.org/wiki/Multi-factor_authentication

After you activate 2FA for a group of users, Keycloak asks those users for their OTP on each login. To simplify the configuration process, you can use a command-line tool to enable 2FA.

To activate or deactivate 2FA for a user group, follow the instructions in the next sections.

3.8.1 Activate two-factor authentication for domain administrators

1. Open a shell on the UCS system as superuser `root` where you installed Keycloak and run the following command:

```
$ univention-keycloak 2fa enable --group-2fa "Domain Admins"
```

2. The next time a user belonging to the `Domain Admins` group tries to sign in, Keycloak forces them to configure the 2FA following the instructions given during the login.

3.8.2 Deactivate two-factor authentication for domain administrators

1. *Sign in to Keycloak Admin Console* (page 4).
2. Navigate to *UCS realm* ▸ *Groups*.
3. Select `Domain Admins` in the list and click *Edit*.
4. Navigate to *Role Mappings* on the tabs.
5. Remove `2FA role` from *Assigned roles*.

3.9 Keycloak ad hoc federation

Warning: The ad hoc federation is a built-in **`Keycloak`** feature that is not integrated into the UCS identity management or user lifecycle. More sophisticated integration needs to be added individually.

New in version 19.0.1-ucs2.

`Keycloak` SPI (Service Provider Interfaces) extension for ad hoc federation added. Keycloak offers identity brokering to delegate authentication to one or more identity providers for OpenID Connect or SAML 2.0.

See also:

For more information about identity brokering and first login flow, see *Keycloak Server Administration Guide: Identity Broker First Login* [6].

The app **`Keycloak`** provides *ad hoc federation* to enable identity brokering and add user accounts to UCS as so-called *shadow accounts*. It supports the *design decision about not having user accounts in Keycloak* (page 30).

The app **`Keycloak`** installs the **`univention-authenticator`** SPI plugin. The plugin creates the local shadow copy of the user account in the OpenLDAP directory services through the REST API of UDM (Univention Directory Manager). *Ad hoc federation* is useful when administrators want to keep track of all users in UCS.

See also:

For more information on SPI, see *Keycloak Server Development Guide: Authentication SPI* [7].

3.9.1 Import external CA certificates

Federation involves other, for example external, server systems and requires trust. Certificates are a way to implement trust. To tell your Keycloak system to trust another system for the ad-hoc federation, you need to import the CA certificate for that system. Keycloak needs the CA certificate to verify the encrypted connection with the other system.

Use the following steps to add the CA certificate of the other system:

```
$ docker cp /path/to/externalCA.pem keycloak:/externalCA.pem
$ univention-app shell keycloak \
keytool -cacerts -import -alias ucsCA -file /externalCA.pem -storepass "changeit" -
↪noprompt
```

Repeat this procedure when any CA certificate expires. In case of any CA related TLS error, restart the container:

```
$ docker restart keycloak
```

3.9.2 Create custom authentication flow

First, you as administrator need to create a custom authentication flow to use *univention-authenticator* SPI:

1. *Sign in to Keycloak Admin Console* (page 4).
2. Navigate to *UCS realm* ‣ *Authentication*.
3. Select `First Broker Login` in the list and click *Copy*.
4. Give a name to the authentication flow and click *OK*.
5. In the *Review Profile (review profile config)* click *Actions* and select `Config`.
6. Select `Off` in the list, click *Save* and navigate back to the authentication flow.
7. Click *Add execution* to get to the *Create Authenticator Execution* page.
8. Select `Univention Authenticator` in the list and click *Save*.
9. On the *Flows* tab in the *Authentication* section, change the *Univention Authenticator* in the displayed table to `Required`.
10. To finish the configuration, click *Actions* in the *Univention Authenticator* and select `Config`.
11. Fill in the following configuration options for the *Univention Authenticator*:

    Alias
    Name of the configuration.

    UDM REST API endpoint
    The API endpoint of UDM where UCS stores the shadow copy of the user.

    Username
    Username of a user account that can write to UDM.

    Password
    Password of the user account that can write to UDM.

12. Click *Save*.

3.9.3 Create an identity provider for Microsoft Active Directory

After you created the *custom authentication flow* (page 15), Keycloak can use ad hoc federation on any configured federated login. In this section, you learn how to set up a federated login using a Microsoft Active Directory Federation Services[12].

To create an identity provider for Active Directory that uses the ad hoc federation follow the next steps:

1. *Sign in to Keycloak Admin Console* (page 4).
2. Navigate to *UCS realm* ▸ *Identity Providers*.
3. Click *Add provider…* and select `SAML v2.0`.
4. Fill in the fields *Alias* and *Display Name*. You **can't** change the field *Alias* later.
5. Select your authentication flow with the *Univention Authenticator* on the *First Login Flow*.
6. Fill in the field *Service Provider Entity ID* with the *EntityID* from the *Relying Party* on the Active Directory Federation Services.
7. Set the *Single Sign-On Service URL* to the single sign-on URL from the *Relying Party*.
8. In *Principal Type* select `Unspecified` in the fields *NameID Policy Format*, *Attribute [Name]*.

   In *Principal Attribute* select `sAMAccountName`.
9. Enable the following properties:
   - `Allow Create`
   - `HTTP-POST Binding Response`
   - `HTTP-POST Binding for AuthnRequest`
   - `Want AuthnRequests Signed`
10. For the field *Signature Algorithm* select `RSA_SHA256`

    For the field *SAML Signature Key Name* select `CERT_SUBJECT`.
11. Enable *Validate Signature* and add the certificate to *Validating x509 Certificates*.
12. Click *Save*

3.9.4 Mappers for the identity provider

The identity provider needs the following mapper configuration to work properly with Univention Corporate Server:

1. To create a mapper in the identity provider configuration navigate to *UCS realm* ▸ *Identity Provider* ▸ *Your Identity Provider* ▸ *Mappers*.
2. Click *Create*
3. Configure the mapper for the email address with the following properties:

   Name
   Name of the mapper

   Sync Mode Override
   `import`

   Type of mapper
   `Attribute Importer`

   Attribute Name
   `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

[12] https://learn.microsoft.com/en-us/windows-server/identity/active-directory-federation-services

User Attribute Name
`email`

4. Configure the mapper for the first name with the following properties:

Name
Name of the mapper

Sync Mode Override
`import`

Type of mapper
`Attribute Importer`

Attribute Name
`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname`

User Attribute Name
`firstName`

5. Configure the mapper for the last name with the following properties:

Name
Name of the mapper

Sync Mode Override
`import`

Type of mapper
`Attribute Importer`

Attribute Name
`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname`

User Attribute Name
`lastName`

6. Configure the mapper for `univentionObjectIdentifier` with the following properties:

Name
Name of the mapper

Sync Mode Override
`import`

Type of mapper
`Attribute Importer`

User attribute
`objectGuid`

User attribute Name
`univentionObjectIdentifier`

7. Configure the mapper for `univentionSourceIAM` with the following properties:

Name
Name of the mapper

Sync Mode Override
`import`

Type of mapper
`Hardcoded attribute`

User attribute
`univentionSourceIAM`

User attribute value
Identifier of the identity provider.

8. Configure the mapper for `external-${ALIAS}-${ATTRIBUTE.sAMAccountName}` with the following properties:

Name
Name of the mapper

Sync Mode Override
`import`

Type of mapper
`Username Template Importer`

User attribute
`external-${ALIAS}-${ATTRIBUTE.sAMAccountName}`

Target
`LOCAL`

3.9.5 Configure Active Directory Federation services for ad hoc federation

To configure the Active Directory Federation Services to properly work with ad hoc federation you need to configure it with the following steps:

1. Sign in as *Administrator* in Active Directory Federation Services.
2. Open *Relying Party Trust* and click *Add Relying Party Trust*.
3. Select `Claim aware` and click *Start*.
4. On the *Select Data Source* page, select `Import data about the relying party published online or on a local network`.
5. In the field *Federation metadata address* insert the metadata URL: `https://ucs-sso-ng.$(ucr get domainname)/auth/realms/ucs/broker/`*`SAML IDP name`*`/endpoint/descriptor`.
6. Specify a *Display Name*. Click *Next*.
7. Select your wanted *Access Control Policy*. Click *Next*.
8. Review your final configuration and click *Next*.
9. Click *Close*.
10. Add the claims to the ticket.

`objectGUID`

1. Click *Add rule* and select `Send LDAP Attributes as Claims`.
2. Add a claim for `objectGUID` to the ticket:

Claim Rule name
Name of the Claim

Attribute Store
`Active Directory`

LDAP attribute
`objectGUID`

Outgoing Claim Type
`objectGUID`

`sAMAccountName`

1. Click *Add rule* and select `Send LDAP Attributes as Claims`.

2. Add a claim for `sAMAccountName` to the ticket:

   Claim Rule name
   Name of the Claim

   Attribute Store
   `Active Directory`

   LDAP attribute
   `SAM-Account-Name`

   Outgoing Claim Type
   `sAMAccountName`

Email address

1. Click *Add rule* and select `Send LDAP Attributes as Claims`.
2. Add a claim for the email address to the ticket:

   Claim Rule name
   Name of the Claim

   Attribute Store
   `Active Directory`

   LDAP attribute
   `E-mail Addresses`

   Outgoing Claim Type
   `E-mail Address`

Given name

1. Click *Add rule* and select `Send LDAP Attributes as Claims`.
2. Add a claim for the given name to the ticket:

   Claim Rule name
   Name of the Claim

   Attribute Store
   `Active Directory`

   LDAP attribute
   `Given-Name`

   Outgoing Claim Type
   `Given Name`

Surname

1. Click *Add rule* and select `Send LDAP Attributes as Claims`.
2. Add a claim for the surname to the ticket:

   Claim Rule name
   Name of the Claim

   Attribute Store
   `Active Directory`

   LDAP attribute
   `Surname`

   Outgoing Claim Type
   `Surname`

11. Apply and save the rules.

3.10 Settings

The following references show the available settings within the **Keycloak** app. Univention recommends to keep the default values.

Keycloak has a lot more possibilities for configuration and customization. For more information, consult *Keycloak 21.1 Documentation* [1].

keycloak/log/level

Configures the verbosity of log messages in Keycloak.

Possible values

ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN.

For a detailed description of the log level values, see *Keycloak documentation: Configuring logging* [8].

Required	Default value	Set
Yes	INFO	Installation and app configuration

keycloak/java/opts

Defines the options that the Keycloak app appends to the *java* command.

Required	Default value	Set
Yes	-server -Xms1024m -Xmx1024m	Installation and app configuration

keycloak/theme

Defines the theme that Keycloak uses for the login interface. A CSS file with the same name must exist in the directory /usr/share/univention-web/themes/. The setting value only uses the basename of the file without the extension css.

Possible values

dark and light

If you provide custom CSS files with other names, they add to the possible values.

Possible values

true and false.

Required	Default value	Set
No	Same value as UCR variable ucs/web/theme[13].	Installation and app configuration

keycloak/server/sso/fqdn

Defines the FQDN to the identity provider in your environment's UCS domain. Defaults to ucs-sso-ng.*$domainname*.

Required	Default value	Set
No	ucs-sso-ng.*$domainname*	Installation and app configuration

keycloak/server/sso/autoregistration

If set to true (default), the UCS system with the Keycloak app installed registers its IP address at the hostname of the identity provider defined in *keycloak/server/sso/fqdn* (page 20).

Possible values:

true or false

[13] https://docs.software-univention.de/manual/5.0/en/appendix/variables.html#envvar-ucs-web-theme

Required	Default value	Set
Yes	`true`	Installation and app configuration

`keycloak/server/sso/virtualhost`

If set to `true` (default) the UCS system will create a dedicated apache virtual host configuration for the Keycloak server FQDN.

Possible values:

`true` or `false`

Required	Default value	Set
Yes	`true`	Installation and app configuration

`keycloak/apache/config`

If set to `true` (default) the UCS system will create an apache configuration for Keycloak.

Possible values:

`true` or `false`

Required	Default value	Set
Yes	`true`	Installation and app configuration

`keycloak/federation/remote/identifier`

This property stores the name of the UDM property that stores the unique identifier of the remote IAM objects. It is only used for ad hoc federation.

Required	Default value	Set
No	`univentionObjectIdentifier`	Installation and app configuration

`keycloak/federation/source/identifier`

This property stores the name of the UDM property that stores the remote source of an IAM objects. It is only used for ad hoc federation.

Required	Default value	Set
No	`univentionSourceIAM`	Installation and app configuration

`keycloak/database/connection`

Specifies the IP addresses from which the default PostgreSQL database can receive connections.

Required	Default value	Set
No	None	Installation and app configuration

`kc/db/url`

Specifies the database JDBC URL (for example `jdbc:postgresql://dbhost/keycloak`) to connect Keycloak. Defaults to `jdbc:postgresql://`*`fqdn`*`:5432/keycloak`.

Required	Default value	Set
No	`jdbc:postgresql://`*`fqdn`*`:5432/keycloak`	Installation and app configuration

kc/db/username

Specifies the database username. Defaults to `keycloak`.

Required	Default value	Set
No	`keycloak`	Installation and app configuration

kc/db/kind

Specifies the kind of database. Defaults to `postgres`.

Required	Default value	Set
No	`postgres`	Installation and app configuration

kc/db/password

Specifies the password to connect to the database.

Required	Default value	Set
No	None	Installation and app configuration

ucs/self/registration/check_email_verification

Controls if the login is denied for unverified, self registered user accounts. For more information, see Account verification[14] in the *UCS 5.0 Manual* [2].

Required	Default value	Set
No	False	Installation and app configuration

keycloak/login/messages/en/accountNotVerifiedMsg

English error message for a self-registered user account that isn't verified yet. The error message supports HTML format.

Required	Default value	Set
No	See default value in Listing 3.1 after the table.	Installation and app configuration

Listing 3.1: Default value for *keycloak/login/messages/en/accountNotVerifiedMsg* (page 22)

```
'Your account is not verified.<br>You must <a id="loginSelfServiceLink" href=
↪"https://${hostname}.${domainname}/univention/selfservice/#/selfservice/
↪verifyaccount" target="_blank">verify your account</a> before you can login.
↪<br/>'
```

keycloak/login/messages/de/accountNotVerifiedMsg

German error message for a self-registered user account that isn't verified yet. The error message supports HTML format.

Required	Default value	Set
No	See default value in Listing 3.2 after the table.	Installation and app configuration

[14] https://docs.software-univention.de/manual/5.0/en/user-management/user-self-service.html#user-management-password-changes-by-users-selfregistration-accou

Listing 3.2: Default value for *keycloak/login/messages/de/accountNotVerifiedMsg* (page 22)

```
'Konto nicht verifiziert.<br>Sie m\\u00FCssen Ihr <a id="loginSelfServiceLink"␣
↪href="https://${hostname}.${domainname}/univention/selfservice/#/selfservice/
↪verifyaccount" target="_blank">Konto verifizieren</a>, bevor Sie sich␣
↪einloggen k\\u00F6nnen.<br/>'
```

keycloak/csp/frame-ancestors

Additional entries to the `frame-ancestors` directive of the Keycloak virtual host. The space separated list of sources can have multiple values can be used. For example, `https://portal.external.com https://*.remote.de`. For more information, see *CSP: frame-ancestors* in Mozilla Foundation [9].

Required	Default value	Set
No	None	Installation and app configuration

keycloak/apache2/ssl/certificate

Sets the absolute path to the SSL certificate file for the **Apache web server** module `mod_ssl` of the Keycloak virtual host. The web server needs the certificate in the PEM format.

The web server uses the UCS certificate from `/etc/univention/ssl/ucs-sso-ng.$domainname/cert.pem`, if the UCR variable has no value.

Required	Default value	Set
No	`/etc/univention/ssl/ucs-sso-ng.$domainname/cert.pem`	Installation and app configuration

keycloak/apache2/ssl/key

Sets the absolute path to the private RSA/DSA key of the SSL certificate file for the **Apache web server** module `mod_ssl` of the Keycloak virtual host. The web server needs the certificate in the PEM format.

The web server uses the UCS private key from `/etc/univention/ssl/ucs-sso-ng.$domainname/private.key`, if the UCR variable has no value.

Required	Default value	Set
No	`/etc/univention/ssl/ucs-sso-ng.$domainname/private.key`	Installation and app configuration

keycloak/apache2/ssl/ca

Sets the absolute path to the certificate of the certificate authority (CA) for the **Apache web server** module `mod_ssl` of the Keycloak virtual host. The web server needs the certificate in the PEM format.

The web server uses the UCS CA from `/etc/univention/ssl/ucsCA/CAcert.pem`, if the UCR variable has no value.

Required	Default value	Set
No	`/etc/univention/ssl/ucsCA/CAcert.pem`	Installation and app configuration

keycloak/cookies/samesite

This setting sets the `SameSite` attribute in all the cookies of Keycloak. Possible values are `Lax`, `Strict` and the default value `None`.

Required	Default value	Set
No	`None`	Installation and app configuration

3.11 Adjusting texts on the Keycloak login page

The **Keycloak** app lets Administrators overwrite any messages on the **Keycloak** login page. Each text variable value in this login template can be overwritten by using a UCR variable of the form

`keycloak/login/messages/[de/en]/key=value`

This make use of the **Keycloak** message bundles that are documented here: https://www.keycloak.org/docs/latest/server_development/#messages

For example, the login title in the **Keycloak** login dialogue can be adjusted like this:

```
$ ucr set \
keycloak/login/messages/en/loginTitleHtml=\
'Login at Domainname'
```

After setting one of these variables, this command has to be run to make the change visible in **Keycloak** login page:

```
$ univention-app configure keycloak
```

Warning: These settings are local settings. The UCR variables have to be set on each host running **Keycloak**.

3.12 Adjusting the Keycloak apache configuration

The **Keycloak** app ships an apache configuration in */etc/apache2/sites-available/univention-keycloak.conf*. This file is created by the app and will be overwritten during updates.

This configuration can be customized by creating the file */var/lib/univention-appcenter/apps/keycloak/data/local-univention-keycloak.conf*

For example, an Administrator may want to restrict the access to the **Keycloak** administration console to a specific IP subnet by putting this in the *local-univention-keycloak.conf*.

```
<LocationMatch "^(/admin/|/realms/master/)">
              deny from all
              allow from 10.207.0.0/16
</LocationMatch>
```

3.13 Activating Kerberos authentication

In the default configuration, the **Keycloak** app evaluates **Kerberos** tickets during the authentication process. If you have a UCS domain with client workstations that obtain **Kerberos** tickets during the user login process, users can configure their web browsers to send this ticket to **Keycloak** for authentication to enable a passwordless login, for example in the UCS portal.

To enable the web browser to send the **Kerberos** tickets, you must change the following settings:

Mozilla Firefox
: Open a new tab and enter `about:config` in the address bar to open the Firefox configuration. Search for `network.negotiate-auth.trusted-uris` and add the FQDN (fully qualified domain name) of your **Keycloak** server, which is `ucs-sso-ng.[Domain name]` by default.

Microsoft Edge

For Microsoft Edge on Windows, you need to configure Kerberos authentication in the general settings of the operating system. Open the *Control Panel* and move to *Security* ‣ *Local Intranet* ‣ *Sites* ‣ *Advanced*. Add the FQDN of your **Keycloak** server, `ucs-sso-ng.[Domain name]` by default, to the list of `Websites`.

If you install the **Active Directory-compatible Domain Controller** app *after* installing **Keycloak**, you need to run the following command on the Primary Directory Node. It ensures that the Kerberos authentication also works with the **Active Directory-compatible Domain Controller**:

```
$ eval "$(ucr shell keycloak/server/sso/fqdn)"
$ samba-tool spn add "HTTP/$keycloak_server_sso_fqdn" "krbkeycloak"
```

Per default, **Keycloak** tries to use **Kerberos**. If no **Kerberos** ticket is available, *Keycloak* falls back to username and password authentication. You can deactivate this behavior in the *Keycloak Admin Console* (page 4) with the following steps:

- Select the realm `UCS`.
- On the sidebar, click *User federation* and choose `ldap-provider`.
- Go to the section *Kerberos integration* and deactivate *Allow Kerberos authentication*.

3.14 Restrict access to applications

New in version 21.1.2-ucs2.

With the UCS **simpleSAMLphp** integration, you can restrict access of groups and users to specific *SAML service providers* through the UDM SAML settings.

The configuration steps in the following sections restrict access to certain *SAML service providers* and *OIDC Relying parties* through group membership in a similar way with **Keycloak**.

Attention: Application access restriction isn't yet integrated into the UDM UMC module yet.

If you already need the application access restriction for groups at this time, read on and follow the steps outlined below. Note that you may need to perform manual migration steps after the integration is complete.

If you don't have an immediate need, it's recommended that you wait until the integration is complete in a future version of the **Keycloak** app.

This configuration differs from the one provided by **simpleSAMLphp** in the following ways:

- Only the group membership restricts the access to applications. It isn't possible to restrict the access for an individual user directly.
- You must configure group access restrictions for *SAML SP* and *OIDC RP* directly in the *Keycloak Admin Console* (page 4), although you manage users and their group memberships in UDM.
- By default, **Keycloak** allows access to all users. Only when you specifically configure the *SAML SP* or *OIDC RP* to use app authorization will **Keycloak** evaluate the access restriction to applications.

3.14.1 Create authentication flow

Keycloak version 21.1.2-ucs2 provides an authenticator extension called *Univention App authenticator*, which performs the authorization validation on the user during the sign-in.

To use this authenticator, you need to create a Keycloak *authentication flow* that includes this authenticator. Use the command **univention-keycloak** as follows. The command doesn't give any output:

Listing 3.3: Create a Keycloak *authentication flow*

```
$ univention-keycloak legacy-authentication-flow create
```

See also:

For more information on authentication flows, see *Keycloak Server Administration Guide: Authentication flows* [10].

3.14.2 Assign authentication flow

Keycloak calls the *SAML SP* and the *OIDC RP Client*. By default, neither *SAML SP* nor *OIDC RP* use the created authentication flow.

To restrict application access, you must assign the *created authentication flow* (page 26) to each *Keycloak Client*. Otherwise, the *Keycloak Client* still allows access to all users. To assign a specific flow to an existing *Keycloak Client*, use the following command in Listing 3.4.

Listing 3.4: Assign authentication flow to a *Keycloak Client*

```
$ univention-keycloak client-auth-flow \
  --clientid "REPLACE_WITH_YOUR_CLIENT_ID" \
  --auth-flow "browser flow with legacy app authorization"
```

Note: You can also pass the option `--auth-browser-flow` when you create a *SAML SP* or *OIDC RP* as a *Keycloak Client*. See section *Keycloak as SAML Identity Provider* (page 11) on how to create a *Keycloak Client*.

3.14.3 Map UDM groups to Keycloak

To restrict access to certain *Keycloak Client*s by group membership, you must map the necessary groups to **Keycloak**. Use the *Keycloak Admin Console* (page 4) to create an appropriate *LDAP mapper*.

1. In *Keycloak Admin Console* (page 4) go to *UCS realm* ‣ *User Federation* ‣ *ldap-provider* ‣ *Mappers* ‣ *Add mapper*.
2. Choose the *Name* of the mapper freely.
3. Select the *Mapper type* `group-ldap-mapper` to extend the form. Fill in the fields as following:

   LDAP Groups DN
   Set to the value of the base LDAP DN of your domain, for example `dc=example,dc=local`.

   Group Object Classes
   `univentionGroup`

   Ignore Missing Groups
   `On`

   Membership LDAP Attribute
   `memberUid`

   Membership Attribute Type
   `UID`

Drop non-existing groups during sync
`On`

Important: It's strongly recommended to set an *LDAP Filter* in the group mapper so that **Keycloak** only maps strictly necessary groups. If you don't specify an *LDAP filter*, **Keycloak** synchronizes **all groups** from the LDAP directory service. Depending on the size of the groups, it may impact the performance of **Keycloak**.

Example
To filter groups by their name and only allow **Keycloak** to synchronize the mentioned groups, use `(|(cn=umcAccess)(cn=nextcloudAccess))`

4. Scroll down and click *Save*.

To trigger the synchronization of the groups immediately, click the name of the mapper you just created to open it and select *Sync LDAP groups to Keycloak* from the *Action* drop-down.

3.14.4 Create client roles

3.14.5 Create Keycloak client roles

The authenticator extension *Univention App authenticator* restricts access by evaluating the roles of a user in **Keycloak**. It specifically checks for a client specific role named `univentionClientAccess`. If this client specific role exists, the authenticator extension restricts access of all users that don't have this role.

For each *Keycloak Client* that you want to check access restrictions, you need to create the role `univention-ClientAccess`. In *Keycloak Admin Console* (page 4) go to *UCS realm ‣ Clients*. For each client of interest, run the following steps:

1. Select *YOUR_CLIENT ‣ Roles ‣ Create role*.
2. Enter name for the role `univentionClientAccess`.
3. Click *Save*.

Important: Follow the next section *Attach the client specific role to groups* (page 27) immediately, because saving the client role enforces the sign-in restriction for the *Keycloak Client*.

See also:

For more information on roles in Keycloak, see *Keycloak Server Administration Guide: Assigning permissions using roles and groups* [11].

3.14.6 Attach the client specific role to groups

To grant access permission to group members of a group so that they can sign in to an app, you need to attach the *Keycloak Client* role to the groups. All group members then inherit the client role.

In *Keycloak Admin Console* (page 4) go to *UCS realm ‣ Groups*. For each group of interest, run the following steps:

1. Select *YOUR_GROUP ‣ Role mapping ‣ Assign role ‣ Filter by clients.*
2. Find and select the app you intend to control with `univentionClientAccess`.

Warning: **Keycloak** doesn't evaluate nested group memberships. Only direct group membership of a user give the user the necessary client role.

3. Click *Assign*.

From now on, only the users that inherited the *Keycloak Client* specific role `univentionClientAccess` have access to the respective applications.

3.14.7 Customize the authorization error page

`Keycloak` shows an error page, if a user doesn't have access to an application because the access restriction applies to them.

You can configure the error page through the following App settings:

German
`keycloak/login/messages/de/accessDeniedMsg`

English
`keycloak/login/messages/en/accessDeniedMsg`

You can include HTML format with links in this setting to customize the error page.

The default message shows the `client ID` of the *Keycloak Client* that forbids access to the user. If you need a human readable name, you can set the attribute *Name* of the *Keycloak Client* in the *Keycloak Admin Console* (page 4). With the attribute set, Keycloak shows the *Name* instead of the `client ID`.

Important: The app setting only applies to the local Keycloak instance. You can use different values on the different Keycloak installations, for example, to show a link to the local portal.

For more information, refer to *Adjusting texts on the Keycloak login page* (page 24).

CHAPTER

FOUR

ARCHITECTURE

The **Keycloak** app architecture consists of the following elements:

- The operating environment UCS with the App Center and the Docker engine running Keycloak.
- The Keycloak software.
- The OpenLDAP LDAP directory in UCS as identity store for Keycloak
- A SQL database as data persistence layer with read-write access for Keycloak.

This architecture view doesn't go into detail of the Keycloak software itself, because it's beyond the scope of this documentation.

4.1 Overview

Fig. 4.1 shows the architecture with the most important elements.

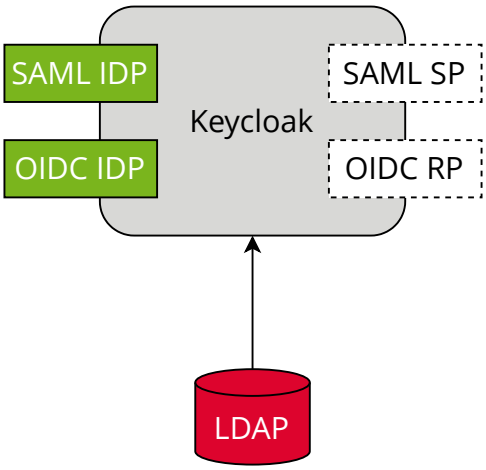

Fig. 4.1: Keycloak app architecture
View focuses on the elements Keycloak, SAML and OIDC as its most important interfaces for single sign-on, and the LDAP directory.

The following list describes the elements in more detail.

Keycloak
: *Keycloak* is the Keycloak software as distributed by the Keycloak project as container image for Docker. The **Keycloak** app uses the unmodified Keycloak binary image and additionally includes files to for example support synchronization of data between instances deployed on the same domain.

LDAP
: *LDAP* is the LDAP directory provided by UCS with the OpenLDAP software. In UCS it is the storage for all

identity and infrastructure data of the UCS domain. For more information, see LDAP directory[15] in *UCS 5.0 Manual* [2].

SAML IDP
SAML IDP stands for *SAML Identity Provider* and is the SAML interface in Keycloak that offers user authentication as a service through SAML.

SAML SP
SAML SP stands for *SAML Service Provider* and is the SAML interface in Keycloak that outsources its user authentication function to an *IDP*.

OIDC Provider
OP is short for *OpenID Connect Provider*. In Keycloak this OIDC interface offers user authentication as a service.

OIDC RP
OIDC RP is short for *OpenID Connect Relying Party*. In Keycloak this OIDC interface outsources its user authentication function to an *OP*.

Keycloak Client
A *Keycloak Client* is any entity that can request Keycloak to authenticate a user. This includes all *OIDC Relying Parties* and *SAML Service Providers*.

4.2 Design decisions

One goal of the **Keycloak** app is to provide a ready to run Keycloak setup for UCS. To reach that goal, the Univention team made the following decisions.

The **Keycloak** app configures a so-called *user federation* in the realm *UCS* in Keycloak. In general, a user federation synchronizes users from LDAP and Active Directory servers to Keycloak. In the Keycloak app, the user federation **doesn't** synchronize user accounts from LDAP to Keycloak, but delegates authentication decisions to LDAP. A realm manages a set of users, credentials, roles, and groups in Keycloak.

The user federation in the realm *UCS* uses the LDAP DN (Distinguished Name) `uid=sys-idp-user,cn=users,$ldap_base` to bind to the LDAP directory in UCS.

The app registers `ucs-sso-ng.$domainname` to the DNS that serves as host for API entry points of Keycloak and administrative web interface.

[15] https://docs.software-univention.de/manual/5.0/en/domain-ldap/ldap-directory.html#domain-ldap

CHAPTER

FIVE

REQUIREMENTS AND LIMITATIONS

To ensure a smooth operation of the **Keycloak** app on UCS, administrators need to know the following requirements and limitations:

5.1 User federation and synchronization

The app configures a user federation in the realm *UCS*. **Don't** remove the user federation or Keycloak won't be able to resolve users anymore.

The configured user federation in the realm *UCS* doesn't synchronize the user accounts from the UCS LDAP to Keycloak. For more information, see *Design decisions* (page 30).

5.2 Installation on UCS

The App Center installs the app **Keycloak** on a UCS 5.0-x Primary Directory Node or Backup Directory Node in your UCS environment, see *Installation* (page 3). The app is suitable for production use in UCS domains. Administrators need to keep in mind, other apps may be unable to authenticate users through SAML without manual reconfiguration.

Administrators need to take care with experiments that involve the reconfiguration, for example, of UMC, and other services to use Keycloak. The experiments may have undesired results. In particular, when you change the UCR variable `umc/saml/idp-server` to point to your Keycloak installation and restart the LDAP server, the LDAP server doesn't accept SAML tickets any longer that the *simpleSAMLphp* based identity provider issued. So users find their existing sessions invalidated.

5.3 No user activation for SAML

In the *Users* UMC module, the user account's *SAML settings* at *Account ▸ SAML settings* don't require anymore that administrators activate identity providers for user accounts. Therefore, any user account can use SAML for single sign-on. The behavior is the same as for the OIDC capability before through the **Kopano Konnect** app.

5.4 Password restriction

Keycloak offers a password policies feature, see *Keycloak Server Administration Guide: Password policies* [12]. Because of the user federation with UCS, see *Design decisions* (page 30), Keycloak **doesn't** manage the users credentials.

UCS takes care of password policy definition and enforcement. For more information, see LDAP directory[16] in *UCS 5.0 Manual* [2].

5.5 Application clients

Keycloak offers the possibility to create SAML or OIDC clients using the command line tool **univention-keycloak**. Administrators can adjust the generic client configuration, if they need a specific configuration. In this case you can use the *Keycloak Admin Console* (page 4).

5.6 Database configuration

By default, the **Keycloak** app uses PostgreSQL as its database back end. The Keycloak app installation procedure automatically installs and configures the database. However, it's not mandatory to use this particular database instance, and administrators may decide to use another one, for example, if there is a need to use an already existing or clustered database.

Important: The *Settings* (page 20) section lists all database settings required for **Keycloak** to work properly. Changing these settings doesn't affect the database itself, no matter if you use the command line tools or the *App Center*. The database settings only tell Keycloak where and how to connect to the database.

Ensure that you first perform the needed changes on the database itself.

---

[16] https://docs.software-univention.de/manual/5.0/en/domain-ldap/ldap-directory.html#domain-ldap

CHAPTER

SIX

USE CASES

This section describes some uses cases for the app **Keycloak** to give a deeper insight of the app's capability.

6.1 Expired password and change password on next sign-in

In some situations, administrators create a user account with a temporary password that requires the account owner to change their password during their first sign-in. The procedure can be company policy or just considered a good practice. Also, if for any other reason like a lost or compromised user password, the account owner can contact the administrator and request a password change.

See also:

User management module - Account tab[17]
For user account expire and set password upon first login, refer to *UCS 5.0 Manual* [2].

To enable these capabilities with **Keycloak**, the app offers the following extensions. The extensions *only* provide the capabilities in the *UCS* realm with the **Keycloak** app installed.

Univention LDAP mapper
In *Keycloak Admin Console* (page 4) follow *UCS realm ‣ User Federation ‣ ldap-provider ‣ Mappers*

The LDAP mapper reads necessary attributes from the LDAP directory and triggers a password update when needed.

Univention update password
In *Keycloak Admin Console* (page 4) follow *UCS realm ‣ Authentication ‣ Required Actions*

Univention update password provides dialogs and forms in the Keycloak login flow.

6.2 Single sign-on through external public domain name

The typical single sign-on configuration in UCS uses a shared DNS record to provide a failover for the sign-in. The *SAML IDP* is available at `ucs-sso-ng.$domainname`. The administrator chose the environment's `$domainname` during UCS installation.

See also:

Domain settings[18] **during UCS installation**
for information about domain modes, settings and naming conventions in the *UCS 5.0 Manual* [2].

Administrators often choose the UCS domain name for an intranet scenario and adapt the service configuration to match the target domain and hostnames. The term FQDN identifies the combination of hostname and domain used to uniquely identify a service. For more information, see Wikipedia contributors [13].

The use case *single sign-on through external, public domain name* addresses administrators who want single sign-on availability from the internet. Administrators find the steps to reconfigure the FQDN for the single sing-on and the

[17] https://docs.software-univention.de/manual/5.0/en/user-management/umc.html#users-management-table-account
[18] https://docs.software-univention.de/manual/5.0/en/installation.html#installation-domain-settings

UCS portal in this section. The configuration for this scenario recommends two UCS servers or more for serving the different FQDNs. If you encounter problems during the steps below, see *Configuration of single sign-on through external public domain* (page 40).

Validate configuration success

Administrators can validate the success of their configuration with the following steps:

1. Use your preferred web browser and open the UCS portal under the just configured FQDN.
2. Sign in as user through single sign-on.
3. After sign-in through single-sign on, the browser redirects you as user back to the UMC portal.
4. If you encounter problems during the validation, see *Configuration of single sign-on through external public domain* (page 40).

Note: The following aspects faced by administrators encounter in this use case are beyond the scope of this document:

- Configuration of an external DNS to point to the UCS system.
- Configuration of network components to route the connection from the internet to the UCS system.
- Obtaining a valid certificate from a CA.

Warning: The **Keycloak** admin interface as well as the **Keycloak** REST API are also publicly available if the **Keycloak** app was configured to be available externally. For security reasons, this should be restricted. Please see *Adjusting the Keycloak apache configuration* (page 24) for an exemplary configuration.

6.2.1 External FQDN different from internal UCS name

New in version 21.0.1-ucs2.

A common scenario is to have the UCS portal available at one FQDN, such as `portal.internet.domain`, and single sign-on available at another different FQDN, such as `sso.internet.domain`.

Before starting with the configuration of this use case, consider the following aspects:

Pre-conditions:

For the scenario described below, it's important to have the following setup in place, before you proceed:

1. You configured the external DNS entry for Keycloak, for example `sso.internet.domain`.
2. You configured the external DNS entry for the UCS portal, for example `portal.internet.domain`.
3. You have obtained proper SSL certificates for Keycloak and the UCS portal new FQDN.

The following steps require a working network access from the UCS system to the external identity provider FQDN.

Configuration of the identity provider

To configure single sign-on on each Keycloak instance in your UCS domain, follow the steps below:

1. Configure the single sign-on FQDN to a custom value. Set the following UCR variables:

```
$ SSO_FQDN=sso.internet.domain
$ ucr set keycloak/server/sso/fqdn="${SSO_FQDN}"
$ ucr set keycloak/server/sso/autoregistration=false
$ ucr set keycloak/apache2/ssl/certificate="/path/to/${SSO_FQDN}/cert.pem"
$ ucr set keycloak/apache2/ssl/key='/path/certificate/${SSO_FQDN}/private.key'
# Add the new public domain of the portal to the frame-ancestor to the CSP
$ ucr set keycloak/csp/frame-ancestors='https://*.internet.domain'

$ univention-app configure keycloak
```

2. Adjust the standard Keycloak portal entry in the UCS domain after changing the single sign-on FQDN:

```
$ udm portals/entry modify \
  --dn "cn=keycloak,cn=entry,cn=portals,cn=univention,$(ucr get ldap/base)" \
  --set link='"en_US "https://sso.internet.domain/admin/"'
```

Warning: After changing the configuration of the identity provider with the previous steps, all services can't use that identity provider until proper configuration.

Configuration of UMC as service provider

To re-configure single sign-on for UMC on all UCS systems in the domain, run the following commands:

```
$ ucr set umc/saml/idp-server="https://${SSO_FQDN}/realms/ucs/protocol/saml/
↪descriptor"
$ service slapd restart
```

For UCS systems joining the domain, configure a UCR policy and assign it the UCS systems before you install them. The UCR policy must set `umc/saml/idp-server` to your custom FQDN.

Configuration of UCS Portal to use external fully qualified domain name

As an example use case to expose the UCS portal to the internet, follow the steps below. Apply the steps only to the UCS system that exposes the UCS portal to the internet.

1. Store the certificate files for the UCS portal in the following locations on your UCS system:
   - Certificate: `/etc/univention/ssl/`*`$PORTAL_FQDN`*`/cert.pem`
   - Private key for the certificate: `/etc/univention/ssl/`*`$PORTAL_FQDN`*`/private.key`
2. Configure the UCR variables to use the custom FQDN and the certificates:

```
$ SSO_FQDN=sso.internet.domain
$ PORTAL_FQDN=portal.internet.domain
$ ucr set umc/saml/sp-server="${PORTAL_FQDN}"
$ ucr set umc/saml/idp-server="https://${SSO_FQDN}/realms/ucs/protocol/saml/
↪descriptor"
```

3. Run the join script to update the web server configuration:

```
$ univention-run-join-scripts \
--force \
--run-scripts 92univention-management-console-web-server.inst
```

6.2.2 External FQDN identical to internal UCS name

New in version 21.1.0-ucs1.

In this scenario the FQDN of the UCS system and the external name for accessing the UCS Portal are identical. Furthermore, the name for the single sign-on endpoint uses the same FQDN. To achieve this use a different URL path for the single sign-on endpoint, for example:

Internal name
`portal.example.test`

External name
`portal.example.test`

Single sign-on URL
`portal.example.test/auth`

Pre-conditions:
For this scenario, it's important to have the following setup in place, before you proceed:

1. You configured the external DNS entry for Keycloak, for example `portal.example.test`.
2. You have obtained proper SSL certificates for this name, for example with the **Let's Encrypt** app from the App Center.

Warning: In this scenario the new **Keycloak** URL path must not be `/` to not override the global configuration of the web server.

Configuration of the identity provider

To configure this scenario run the following steps on each **Keycloak** instance in your UCS domain.

```
$ FQDN="portal.example.test"
$ SSO_PATH="/auth"
$ ucr set keycloak/server/sso/fqdn="$FQDN"
$ ucr set keycloak/server/sso/path="$SSO_PATH"
$ ucr set keycloak/server/sso/virtualhost=false
$ ucr set keycloak/server/sso/autoregistration=false

$ univention-app configure keycloak
```

Warning: After changing the configuration of the identity provider with the previous steps, all services can't use that identity provider until proper configuration.

Configuration of UMC as service provider

To re-configure single sign-on for UMC on all UCS systems in the domain, run the following commands:

```
$ FQDN="portal.example.test"
$ SSO_PATH="/auth"
$ ucr set \
  umc/saml/idp-server="https://${FQDN}${SSO_PATH}/realms/ucs/protocol/saml/
↪descriptor"
$ service slapd restart
```

For UCS systems joining the domain, configure a UCR policy and assign it the UCS systems before you install them. The UCR policy must set `umc/saml/idp-server` to your custom *SAML IDP* URL.

6.2.3 Official (Let's Encrypt) certificates for single sing-on

If the single sing-on endpoint is exposed to the internet, usually an official certificate for the server is required. This can be achieved with the **Let's Encrypt** app (but it is not required to use this app to create the official certificate).

Note: The examples below assume the **Let's Encrypt** was used to create the certificate. The actual filenames of the certificate and key can differ depending on which mechanism was used to create the certificate.

Dedicated FQDN for single sing-on endpoint

In case you use the **Let's Encrypt** app, you have to configure **Let's Encrypt** to acquire a certificate for both names, the UCS portal and **Keycloak**. Apply the following app settings for the **Let's Encrypt** app:

App setting	Value
Domain(s) to obtain a certificate for, separated by space	`portal.extern.com auth.extern.com`
Use certificate in Apache	`yes`

In this scenario the single sing-on endpoint has its own web server configuration, as web server virtual host. To configure the certificate files for **Keycloak**, set the following UCR variables:

Listing 6.1: Set UCR variables to use the Let's Encrypt certificate

```
$ cert_file="/etc/univention/letsencrypt/signed_chain.crt"
$ key_file="/etc/univention/letsencrypt/domain.key"
$ ucr set keycloak/apache2/ssl/certificate="$cert_file"
$ ucr set keycloak/apache2/ssl/key="$key_file"
$ systemctl reload apache2.service
```

Single sing-on FQDN identical to UCS Portal FQDN (or internal name)

If you use the **Let's Encrypt** app to generate the certificates, you need the following app settings for **Let's Encrypt**:

App Setting	Value
Domain(s) to obtain a certificate for, separated by space	`portal.extern.com`
Use certificate in Apache	`yes`

In this use case, the **Keycloak** app uses the global web server configuration. You can therefore use the standard UCR variables for the **Apache** certificate files as outlined in Listing 6.2.

Listing 6.2: Assign certificates to web server configuration

```
$ cert_file="/etc/univention/letsencrypt/signed_chain.crt"
$ key_file="/etc/univention/letsencrypt/domain.key"
$ ucr set apache2/ssl/certificate="$cert_file"
$ ucr set apache2/ssl/key="$key_file"
$ systemctl reload apache2.service
```

CHAPTER

SEVEN

TROUBLESHOOTING

When you encounter problems with the operation of the **Keycloak** app, this chapter provides information where you can look closer into and to get an impression about what is going wrong.

7.1 Log files

The **Keycloak** app produces different logging information in different places.

/var/log/univention/appcenter.log
: Contains log information around activities in the App Center.

  The App Center writes Keycloak relevant information to this file, when you run app lifecycle tasks like install, update and uninstall or when you change the app settings.

/var/log/univention/join.log
: Contains log information from join processes. When the App Center installs Keycloak, the app also joins the domain.

Keycloak Docker container
: The app uses the vanilla Keycloak Docker image[19]. The App Center runs the container. You can view log information from the Keycloak Docker container with the following command:

```
$ univention-app logs keycloak
```

Keycloak Admin Console
: Offers to view event logs in *Events* in the *Manage* section. Administrators can see *Login Events* and *Admin Events*. For more information, see *Keycloak Server Administration Guide: Configuring auditing to track events* [14].

7.2 Debugging

To increase the log level for more log information for the **Keycloak** app, see *keycloak/log/level* (page 20).

This log level only affects the log information that Keycloak itself generates and writes to the Docker logs. The App Center sets the Docker container's `KEYCLOAK_LOGLEVEL` environment variable to the value of *keycloak/log/level* (page 20).

[19] https://quay.io/repository/keycloak/keycloak

7.3 Configuration of single sign-on through external public domain

Administrators may encounter some problems when reconfiguring of the Univention Management Console and Keycloak for a custom FQDN. This section describes the most common problems that may occur.

7.3.1 Univention Management Console join script failure

During the run of the UMC (Univention Management Console) join script as described in *Configuration of UMC as service provider* (page 35), the join script may fail with the error code `3`.

During the script run, the join script downloads the SAML metadata from the *SAML IDP* specified in `umc/saml/idp-server`. The download was unsuccessful. Check manually, for example with your web browser, if you can reach the metadata at `https://$SSO_FQDN/realms/ucs/protocol/saml/descriptor`. After you can load the metadata manually, run the following commands:

```
# Set the SAML metadata url
$ ucr set umc/saml/idp-server="https://${SSO_FQDN}/realms/ucs/protocol/saml/
↪descriptor"

# Execute the join script again
$ univention-run-join-scripts --force --run-scripts 92univention-management-
↪console-web-server.inst
```

7.3.2 Single sign-on session not refreshed

After a sign-in to the UCS portal through single sign-on, the portal passively refreshes the user session every five minutes. If the configuration of the Keycloak virtual host in the Apache web server is incorrect, the passive refresh doesn't work for the UCS portal or other services.

To allow external connections to Keycloak, you need to add the sources as space separated list to the UCR variable *keycloak/csp/frame-ancestors* (page 23).

Tip:

Recommendation
To test this behavior, use a private or incognito session in your web browser.

CHAPTER

EIGHT

CHANGELOG

This changelog documents all notable changes to the **Keycloak** app. Keep a Changelog[20] is the format and this project adheres to Semantic Versioning[21].

Please also consider the upstream release notes[22].

8.1 21.1.2-ucs2

Released: 18. August 2023

- The app can now be configured to restrict access to certain apps using group memberships. For more information about the configuration of this feature, see *Restrict access to applications* (page 25).
- If the *Keycloak* hostname is accessed using http, you are now directly redirected to https
- Due to longer replication times during password updates, it could happen that after a successful password update during the *Keycloak* login an error was shown. This has been fixed.

8.2 21.1.2-ucs1

Released: 19. July 2023

- The app updates to *Keycloak* version 21.1.2 of the upstream Docker image from https://quay.io/repository/keycloak/keycloak.

8.3 21.1.1-ucs1

Released: 5. July 2023

- The app updates to *Keycloak* version 21.1.1 of the upstream Docker image from https://quay.io/repository/keycloak/keycloak. See release notes for Keycloak 21.1.0[23] for more details.
- The app now configures **Kerberos** ticket authentication through the web browser. For more information, see *Activating Kerberos authentication* (page 24).

[20] https://keepachangelog.com/en/1.0.0/
[21] https://semver.org/spec/v2.0.0.html
[22] https://www.keycloak.org/docs/latest/release_notes/index.html
[23] https://www.keycloak.org/docs/latest/release_notes/index.html#keycloak-21-1-0

8.4 21.0.1-ucs4

Released: 28. June 2023

- A Base64 *NameID* mapper has been added, to make the migration of the Microsoft365 connector to **Keycloak** possible.

8.5 21.0.1-ucs3

Released: 31. May 2023

- The UCR variable *keycloak/apache/config* (page 21) replaces the variable `ucs/server/sso/virtualhost`. In case you set `ucs/server/sso/virtualhost` to `false` to turn off the UCS web server configuration for **Keycloak**, set *keycloak/apache/config* (page 21) to `true` before the update.
- The app can use a different URL path for the single sign-on endpoint. For more information about the configuration, see *Single sign-on through external public domain name* (page 33).

8.6 21.0.1-ucs2

Released: 28. April 2023

- The **Keycloak** app can use an external fully qualified domain name. For more information about the configuration, see *Single sign-on through external public domain name* (page 33).

8.7 21.0.1-ucs1

Released: 19. April 2023

- From this version on the **Keycloak** app requires a CPU that supports the micro architecture level `x86-64-v2`. For more information, see Univention Help 21420[24].
- The app updates *Keycloak* to version 21.0.1 of the upstream Docker image from keycloak / keycloak - Quay[25]. See release notes for Keycloak 21.0.0[26] for more details.
- Accessing the `userinfo` endpoint now requires inclusion of `openid` in the list of requested scopes. For background information, see this upstream issue[27].

8.8 19.0.2-ucs2

Released: 23. March 2023

- This release of the **Keycloak** app includes extensions for
  1. Univention LDAP mapper
  2. Univention Password reset
  3. Univention Self service

[24] https://help.univention.com/t/21420
[25] https://quay.io/repository/keycloak/keycloak
[26] https://www.keycloak.org/docs/latest/release_notes/index.html#keycloak-21-0-0
[27] https://github.com/keycloak/keycloak/issues/14184

- **Keycloak** now checks the password expiry during the sign-in and presents a password change dialog if the password has expired.
- The app now offers a setting to deny the sign-in for unverified, self registered user accounts. For more information, see *use cases* (page 33).

8.9 19.0.1-ucs3

Released: 14. October 2022

- This release of the **Keycloak** app includes an extended version of the command line program **univention-keycloak**. Use it to directly create Keycloak *Client* configurations for *SAML Service Providers* and *OpenID Connect Relying Parties*.

8.10 19.0.1-ucs2

Released: 9. September 2022

- This release of the **Keycloak** app includes an SPI extension for so called ad-hoc federation. See the documentation for details.
- Administrators can install the app **Keycloak** on UCS 5.0-x UCS Primary Directory Nodes. For more information, see *Installation on UCS* (page 31).

8.11 19.0.1-ucs1

Released: 7. September 2022

- The app now offers **univention-keycloak**, a command line program to configure *SAML SP* and *OIDC Provider* clients in *Keycloak* directly.

  univention-keycloak simplifies the integration of client apps with *Keycloak* and the downloads of signing certificates for example as PEM file (see option groups `saml/idp/cert` or `oidc/op/cert`).
- **univention-keycloak** supports the setup of a 2FA authentication flow for the members of a specific LDAP group. The second factor is a time-based one-time password (TOTP) in this case.
- The app updates to *Keycloak* version 19.0.1 of the upstream Docker image from https://quay.io/repository/keycloak/keycloak.
- Administrators can install the app **Keycloak** on UCS 5.0-x UCS Primary Directory Nodes. For more information, see *Installation on UCS* (page 31).

8.12 18.0.0-ucs1

Released: 28. June 2022

- Initial release of the app.
- Administrators can install the **Keycloak** app on UCS 5.0-x Primary Directory Nodes.
- The app uses the upstream Docker image from https://quay.io/repository/keycloak/keycloak.

CHAPTER

NINE

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] *Keycloak 21.1 Documentation*. URL: https://www.keycloak.org/archive/documentation-21.1.html.

[2] *UCS 5.0 Manual*. Univention GmbH, 2021. URL: https://docs.software-univention.de/manual/5.0/en/.

[3] Raphaël Hertzog and Roland Mas. *The Debian Administrator's Handbook*, chapter Shell and Basic Commands. Freexian SARL, First edition, 2020. URL: https://www.debian.org/doc/manuals/debian-handbook/short-remedial-course.en.html#sect.shell-and-basic-commands.

[4] *Keycloak Server Administration Guide: OIDC clients*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#_oidc_clients.

[5] *Keycloak Server Administration Guide: Creating a SAML client*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#_client-saml-configuration.

[6] *Keycloak Server Administration Guide: Identity Broker First Login*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#_identity_broker_first_login.

[7] *Keycloak Server Development Guide: Authentication SPI*. URL: https://www.keycloak.org/docs/21.1.2/server_development/#_auth_spi.

[8] *Keycloak documentation: Configuring logging*. URL: https://www.keycloak.org/server/logging#_configuring_the_root_log_level.

[9] Mozilla Foundation. CSP: frame-ancestors. In *Web technology for developers*, chapter CSP: frame-ancestors. Mozilla Foundation, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors.

[10] *Keycloak Server Administration Guide: Authentication flows*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#_authentication-flows.

[11] *Keycloak Server Administration Guide: Assigning permissions using roles and groups*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#assigning-permissions-using-roles-and-groups.

[12] *Keycloak Server Administration Guide: Password policies*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/#_password-policies.

[13] Wikipedia contributors. Fully qualified domain name — Wikipedia, the free encyclopedia. 2023. [Online; accessed 25-April-2023]. URL: https://en.wikipedia.org/w/index.php?title=Fully_qualified_domain_name&oldid=1146805131.

[14] *Keycloak Server Administration Guide: Configuring auditing to track events*. URL: https://www.keycloak.org/docs/21.1.2/server_admin/index.html#configuring-auditing-to-track-events.

INDEX