ID Broker manual for school authorities

Univention GmbH

Mar 07, 2024

CONTENTS:

This documentation is intended for administrators and stakeholders of *school authorities* who want to connect their UCS@school instance to the ID Broker.

As a *school authority*, you can use the ID Broker to provide single sign-on (SSO) to end users between the *identity provider (IDP)* of connected *school authorities* and services. The ID Broker connects IDP and school authority. Service specific pseudonyms prevent user profiles based on combined user activities.

CHAPTER

ONE

BIG PICTURE OF UNIVENTION ID BROKER

The Univention ID Broker eases the integration between identities of learners and teachers managed by *school authorities* or federal states and the various *service providers* for educational purposes with respect to the data protection regulations in Europe[1].

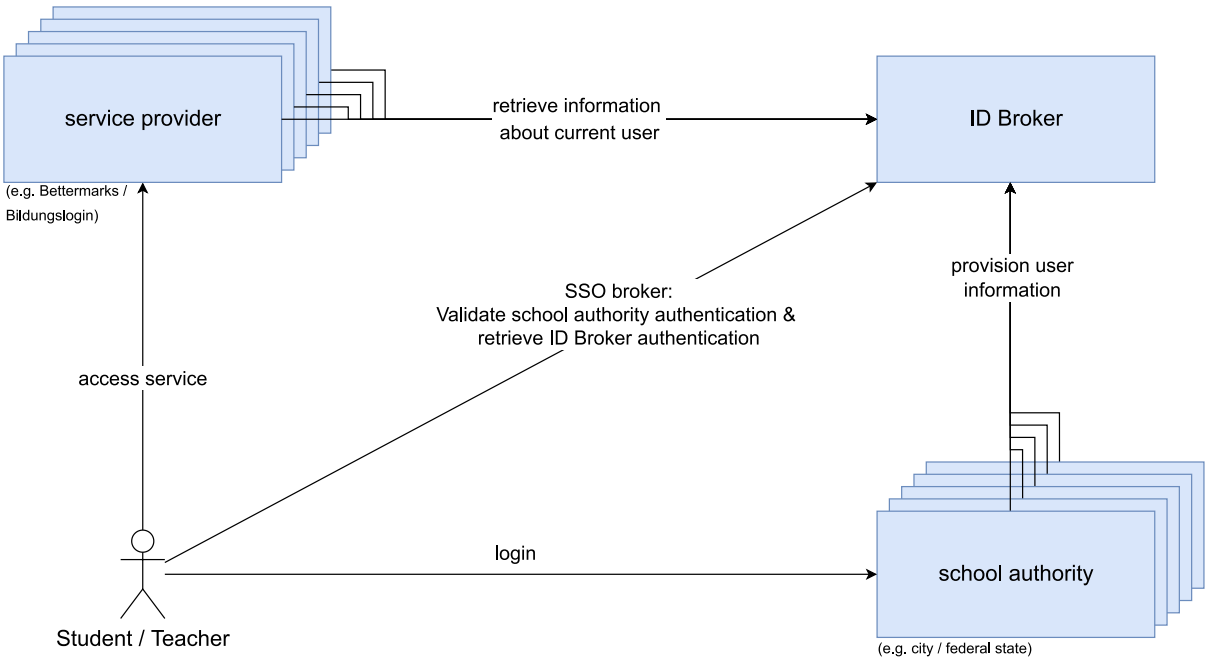

Figure 1.1: Overview of the involved components of the ID Broker and external Systems.

To reach this goal the ID Broker ensures the following:

- Single sign-on for end users between the *IDP* of a *school authority's IDP* and *service providers* (educational SaaS offerings).
- Only one configuration step to connect with the ID Broker both for IDPs and service provider. There is no need to configure each IDP with each service.
- User identification uses service specific pseudonyms instead of global identifiers. Service specific pseudonyms prevent user profiles based on combined user activities in the different services.
- To give end users a *complete* environment from scratch, *service providers* can retrieve information about the role and the courses of users.
- To ensure data protection, the ID Broker environment only stores the user's first name, last name, email address as well as the school class and school memberships.

[1] https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

The UCS@school components of the ID Broker, like the UCS@school Kelvin REST API, are built on top of UCS core components OpenLDAP, UDM and the UDM REST API. To learn more about UCS and it's components, see Univention documentation[2].

[2] https://docs.software-univention.de

CHAPTER

TWO

IN-DEPTH

In section *Authentication and user data retrieval* (page 5) we have a look at the steps, that users follow when they access a service registered at the ID Broker. The section *SSO Broker* (page 7) covers the role and access points of the SSO Broker.

2.1 Authentication and user data retrieval

One design goal of the ID Broker architecture is that the users of multiple *school authorities* can securely access the resources of multiple service providers, so that the *school authorities* and the *service providers* don't have to communicate with each other. Users only login at their *school authority*. The *service providers* don't store any user information.

When a user wants to access a resource of one of the *service providers*, they need to authenticate themselves. The *service provider* requires some data about the users to provide an individualized service.

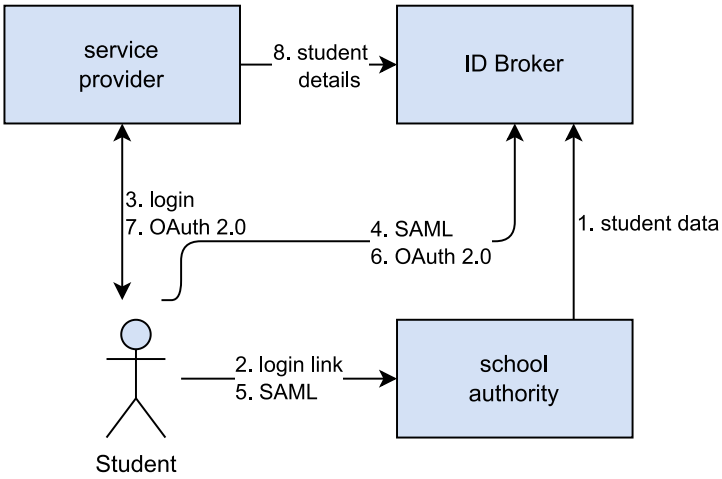

Figure 2.1: ID Broker - connections

Figure 2.1 above shows the connections between a student, the *service provider*, the *school authority* and the ID Broker. It redirects the user to the ID Broker, which in turn redirects the user to the *IDP* of its school authority.

The ID Broker verifies the signature of the *school authorities IDP* and gives a ticket to the user. The user passes that ticket to the *service provider*, which can now retrieve data about the user from the ID Broker.

The following Figure 2.2 covers the interactions between the components in more depth.

1. student data
The *school authority* syncs student data to the ID Broker.

2. request service provider login at school authorities portal page
The student clicks a link on the *school authorities* portal page, and is redirected to the *service provider*. This

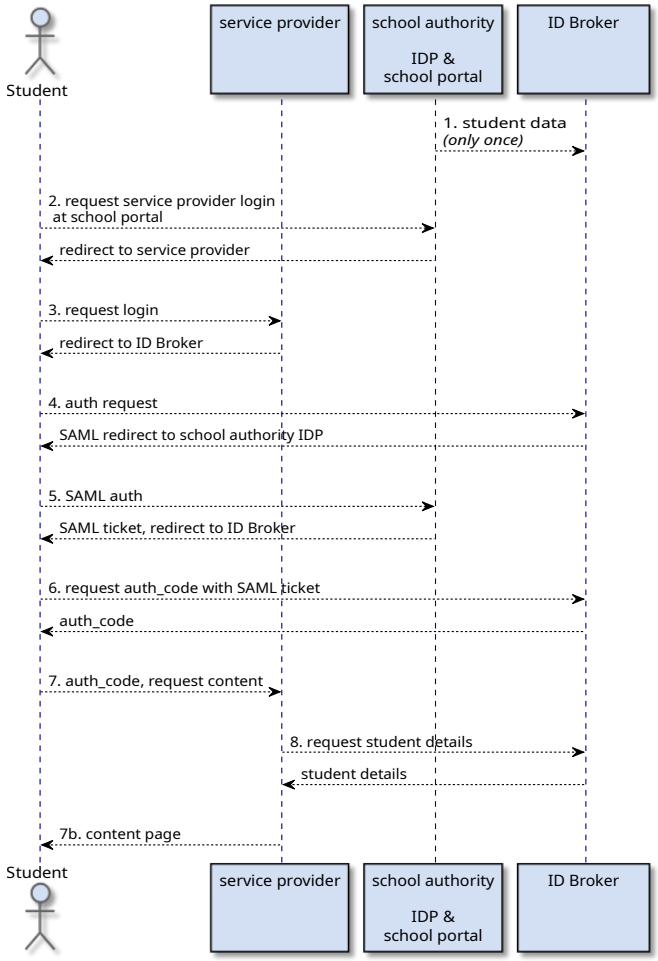

Figure 2.2: ID Broker sequence: authentication and user data retrieval sequence

redirection makes the combination of SAML and OpenID Connect possible - the ID Broker must know which SAML backend needs to be used.

3. request login
The student requests a login at the *service provider's* page, and is redirected to the ID Broker.

4. auth request
The ID Broker doesn't login the student and instead redirects the student to the *school authority*, which has an *IDP* (SAML) provider.

5. SAML auth
The actual SAML authentication of the user happens. The student receives a SAML ticket and is redirected to the ID Broker.

6. request `auth_code`
Using the SAML ticket, the user requests an `auth_code` from the ID Broker. The user is redirected to the *service provider*.

7. `auth_code`, request content
The user passes the `auth_code` to the *service provider* while asking for the content. The service provider exchanges the `auth_code` for an `access_token` and an `id_token` (this step is left out of the diagram for clarity reasons). The `id_token` contains the pseudonyms for the requested data, as well as, for the requesting user.

8. request student details
Using the `access_token` and the pseudonyms inside the `id_token`, the *service provider* can now request pseudonymized user data from the ID Broker.

7b. content page
This is the continuation of step 7 - the student receives the requested content from the *service provider*.

2.2 SSO Broker

In this section you learn about the architecture with focus on the SSO Broker and the single sign-on part of the ID Broker. The software Keycloak provides the functionality for the SSO Broker.

The main job of the SSO Broker component is to handle multiple-tenant authentication, using pseudonyms. This involves the student doing the login and passing authentication tokens back and forth.

The *SSO Broker* participates in the following communications:

- The school portal redirects the student to the SSO Broker upon first login. This first step is part of the OpenID Connect (OIDC) flow. The SSO Broker redirects the student to the *school authority's IDP* for SAML authentication. The student authenticates with a real user identifier. The student returns the SAML ticket to the SSO Broker, which they received in the authentication step.
- The ID Broker IDM system provides a service provider specific pseudonym for the SSO Broker. The pseudonym also includes other user data from the *school authority*. The student receives an `auth_code` that is valid for the *service provider* specific pseudonym. The student sends the `auth_code` to the *service provider*.
- The *service provider* exchanges the `auth_code` for both an `access_token` and an `id_token` at the SSO Broker. The *service provider* processes the `id_token` that contains the pseudonym. It uses the `access_token` to request more data about the student through the pseudonym at the *Self-Disclosure API*.

The SSO Broker is available:

- for OIDC at `https://FQDN/auth/realms/SERVICE PROVIDER ID/protocol/openid-connect`
- for SAML at `https://FQDN/auth/realms/SERVICE PROVIDER ID/broker/saml`

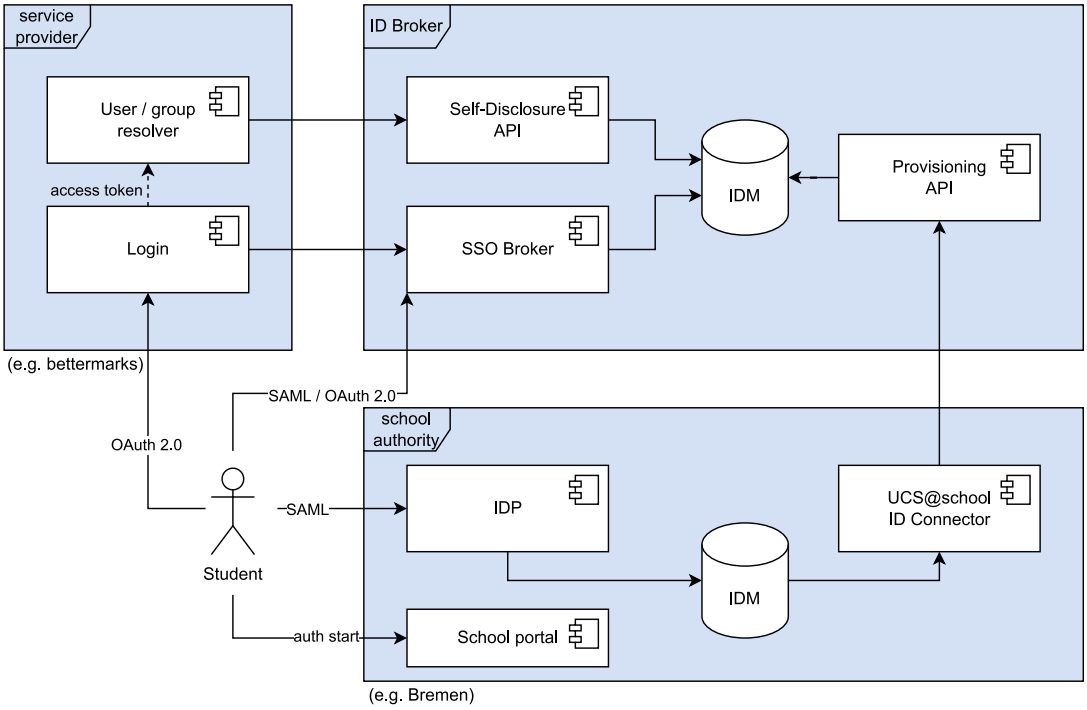

Figure 2.3: SSO Broker communications

CHAPTER

THREE

INSTALLATION

The ID Broker requires the installation of the UCS app **UCS@school ID Connector**. The UCS@school ID Connector offers the possibility to connect a UCS@school domain to another UCS@school domain and to provision it with user data. In this case the source is the UCS@school domain of the *school authority* and the target is the ID Broker system.

For more information about the UCS@school ID connector, see the UCS@school ID connector documentation[3].

To use the UCS@school ID Connector in conjunction with the ID Broker you have to install the *ID Connector Plugin*. It uses an API client to create users, groups and OU objects on the ID Broker system. The UCS@school ID Connector creates schools that are not yet synchronized, after the first data change in the *school authority* system.

3.1 Installation on school authority systems

Prerequisite for the installation and configuration is a UCS@school domain with an already configured **UCS@school** app. If you need information about the setup of a UCS@school domain, have a look to the Quickstart Guide for UCS@school[4] as well to the Manual for Administrators[5].

Another requirement relates to the app **UCS@school ID Connector** itself. As administrator you can only install the app on UCS@school systems with the system roles *Primary Directory Node* or *Backup Directory Node*. This way the UCS@school ID Connector, which is synchronizing data to the ID Broker system, can be used next to an already existing UCS@school ID Connector on the *Primary Directory Node* which synchronizes data to another target.

If you consider multiple systems with a matching system role for installation, keep the following information about the expected system load in mind:

- The ID Connector generates a moderate system load during initial provisioning and during school year change.
- The ID Connector generates low system load during LDAP changes in production operation.

The following steps describe how to install the required components for the ID Connector in your UCS@school domain. Run the commands as user `root` on the console. For details about the configuration of the ID Connector, see section *Configuration* (page 11).

1. Install the **UCS@school ID Connector** app. The app version must be later than `2.3.2`:

```
$ univention-app install ucsschool-id-connector
```

2. Install the ID Broker plugin for the ID Connector:

```
$ univention-install id-broker-id-connector-plugin
$ univention-app restart ucsschool-id-connector
$ service univention-appcenter-listener-converter@ucsschool-id-connector stop
$ find /var/lib/univention-appcenter/listener/ucsschool-id-connector/ \
```

(continues on next page)

[3] https://docs.software-univention.de/ucsschool-id-connector/
[4] https://docs.software-univention.de/quickstart-ucsschool-5.0-de.html
[5] https://docs.software-univention.de/ucsschool-handbuch-5.0.html

(continued from previous page)

```
> /var/lib/univention-appcenter/apps/ucsschool-id-connector/data/listener/ -
↪type f -delete
```

After the installation you can access the API documentation at: `https://[FQDN]/ucsschool-id-connector/api/v1/docs`. Replace `FQDN` with the fully qualified domain name of your UCS system that has the **`UCS@school ID Connector`** app installed.

Note: The API is accessible by default on the UCS system and requires authentication upon access to the API endpoints. Nevertheless, we recommend that you do not make the API available directly from the Internet.

Note: To use all features described in **`id-broker-id-connector-plugin`**, update to the newest version of the debian package **`id-broker-id-connector-plugin`**.

CHAPTER

FOUR

CONFIGURATION

The **UCS@school ID Connector** app offers the possibility to connect a UCS@school domain to another UCS@school domain and to provision it with user data. Plugins can extend the functionality of the **UCS@school ID Connector**. Depending on the needed API on the target system, a specific plugin is required.

If you completed all the steps from the *Installation* (page 9) section, you already installed the necessary plugin for the connection to the ID Broker system on the UCS system.

In the following, you will now first learn about the configuration of the plugin and then the configuration of the *Identity Provider (IDP)* in your UCS@school domain.

As administrator of a *school authority* contact the operator of the ID Broker at school-authority-admin@univention-id-broker.com to start the registration process. Your registration request has to include the name of the school authority and the public FQDN of the school authority's SAML *IDP*.

For the configuration of your system, the ID Broker team provides some values that are required for the configuration. The single configuration steps explain the respective values in detail. The values include a namespace ID, a username, and the corresponding password. In the following examples for commands and file contents, you must replace the corresponding placeholders (*$NAMESPACEID*, *$USERNAME* and *$PASSWORD*) with the concrete value that were given to you.

4.1 UCS@school ID Connector Plugin configuration

Changes in the *school authority's* IDM (LDAP directory) trigger the UCS@school ID Connector plugin for the **UCS@school ID Connector**. Changes can be, for example, the creation, modification, and deletion of UCS@school users and school groups.

Upon such a change in the IDM, the plugin uses the ID Broker's *Provisioning API* to create, modify or delete user and group data on the ID Broker system accordingly. If an object is part of a school, that doesn't yet exist on the ID Broker, the plugin creates the corresponding school on the ID Broker automatically.

To access the *Provisioning API* of the ID Broker, the plugin requires a *namespace ID*, a *username* and the corresponding *password* in its configuration:

- The *namespace ID* is an ID that represents your *school authority* domain in the ID Broker system. All objects like users, school groups and schools that are provisioned to the ID Broker also contain the *namespace ID*, which makes the objects uniquely assignable to your system.
- The ID Broker's *Provisioning API* requires an authenticated access. The user account created for you has the appropriate permission for the API to create, modify or delete objects with your *namespace ID* only.

The username usually consists of the prefix `provisioning-` and the namespace ID. For example, a *school authority* with the namespace ID `ExampleSchoolAuthority` receives the username `provisioning-ExampleSchoolAuthority` from the ID Broker team.

Please perform the following actions once to configure the UCS@school ID Connector plugin:

1. The configuration file of the plugin requires the JSON format. Create the file `school_authority.json`, for example `/root/school_authority.json`, in a directory of your choice on the UCS@school

ID Connector system. Remember, you must replace the placeholders `$NAMESPACEID`, `$USERNAME` and `$PASSWORD` with the appropriate values. The file has to contain the following content:

```
{
    "name": "$NAMESPACEID",
    "active": true,
    "url": "https://provisioning.production.univention-id-broker.com/",
    "plugins": ["id_broker-users", "id_broker-groups"],
    "plugin_configs": {
        "id_broker": {
            "password": "$PASSWORD",
            "username": "$USERNAME",
            "version": 1,
        }
    }
}
```

2. Use the following two commands to send the JSON configuration to an API of the **UCS@school ID Connector**. This API also allows only authenticated access. Therefore, before calling it, replace the placeholder `$ADMINPW` with the password of the user `Administrator` from your domain. Run the adapted commands on the UCS system that has the app **UCS@school ID Connector** installed and where the file `school_authority.json` exists.

```
$ token=$(curl -X POST \
> https://$(hostname -f)/ucsschool-id-connector/api/token \
> -H "Content-Type:application/x-www-form-urlencoded" \
> --data-urlencode "username=Administrator" \
> --data-urlencode "password=$ADMINPW" | \
>  python -c 'import json,sys;print json.load(sys.stdin)["access_token"]')

$ curl -X POST https://$(hostname -f)/ucsschool-id-connector/api/v1/school_
↪authorities \
> -H "Content-Type:application/json" \
> -H "Authorization: Bearer $token" \
> --data-binary @school_authority.json
```

3. After you have successfully uploaded the JSON configuration, you can check the connection to the ID Broker's *Provisioning API* with the **initial_sync.py** tool.

```
$ univention-app shell ucsschool-id-connector \
> /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/initial_sync.py --dry-mode
```

After the **UCS@school ID Connector** is configured, you can **connect one or more new schools to the ID Broker** with the **manage_schools_to_sync.py** tool.

Note: If the school authority has been configured already with an ID Broker plugin prior to version `1.3.16` the existing behavior is kept, meaning all existing and future schools are synchronized.

The following commands have to be executed to add the schools *ou1* and *ou2* to the ID Broker, which are not yet connected to the ID Broker.

```
$ systemctl stop univention-appcenter-listener-converter@ucsschool-id-connector.
↪service
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py add_schools --school_authority Traeger2 "ou1" "ou2"
$ systemctl start univention-appcenter-listener-converter@ucsschool-id-connector.
↪service
```

This will synchronize all users without groups of `ou1` and `ou2`, then synchronize all groups without members and finally adapt all group memberships. You can use `"*"` as a wildcard character in your school names. When the script is finished, all following user & group updates for `ou1` and `ou2` and all other schools which where configures earlier will be synced to the ID Broker. To handle errors that happen during synchronization, please refer to section *Error handling for manage_schools_to_sync.py* (page 16). The service `univention-appcenter-listener-converter@ucsschool-id-connector.service` is paused because during the initial synchronization the **`UCS@school ID Connector`** operates in a special mode not suitable for normal operation.

You can add all existing schools using the flag `--all_schools`. With this option, all future schools need to be added manually. Alternatively you can pass `"*"` as the `SCHOOLS` argument to add all existing and future schools use to the ID Broker. Use both options with care as they will add all schools to the ID Broker.

For more information refer to section *Error handling for manage_schools_to_sync.py* (page 16).

Note: Please note that the synchronization time can vary greatly depending on the number of users and the nature of the group memberships. As a rule of thumb, an initial synchronization duration of 6-7 days can be assumed for approximately 85,000 user accounts and 25,000 school groups. Consider opening a **`screen`** session.

Schools can be **removed from the ID Broker** by using the same tool.

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py remove_schools --school_authority Traeger2 "ou1" "ou2
↪"
```

This will remove the schools `ou1` and `ou2` from the school authority configuration and remove them from the ID Broker.

To remove a school name from the configuration, it needs to match exactly. The argument `"ou*"` will only match `"ou*"` in the configuration not `"ou1"`. If `"ou*"` can be removed from the configuration, all matching schools will be removed from the ID Broker. Matching schools will not be removed if they still match with another school name in the configuration. If you previously added `"ou1"` and `"*"` to the configuration and remove `"*"`, `"ou1"` will not be deleted since it is still configured to be synced.

Alternative to removing the schools by name, you can use the flag `--all_schools` to remove all schools which were to the school authority configuration from the school authority configuration and the ID Broker system. Use this option with care.

Note: In case you are familiar with the configuration of the **`UCS@school ID Connector`**: A school to *authority* mapping is not needed since all schools are synced to the ID Broker. You will find the connected schools in the `school_authority.json`.

4.2 Login with SSO using the ID Broker

Once you configured the **`UCS@school ID Connector`** to actively provision users and groups to the ID Broker system, you need to setup a trust context between the *IDP* of the *school authority* and the ID Broker system to enable the login for a school through SSO with the ID Broker.

Run the following steps on the UCS system with the **`UCS@school ID Connector`** app installed as user `root`:

1. The SAML assertion issued by your *IDP* must also contain the `entryUUID` attribute to work with the ID Broker. Use the following command to add this attribute to the SAML assertion:

```
$ udm saml/idpconfig modify \
> --dn "id=default-saml-idp,cn=univention,$(ucr get ldap/base)" \
> --append "LdapGetAttributes=entryUUID"
```

2. To activate automatic variable substitution in all of the following commands, set the environment variable `NAMESPACEID`. Replace in the following example `ExampleSchoolAuthority` with the corresponding value.

```
$ export NAMESPACEID="ExampleSchoolAuthority"
```

3. Download the configured metadata for this *school authority* from the authentication service called **Keycloak**, that runs on the ID Broker system. Run the following command to download the metadata and store it in the file `metadata.xml`:

```
$ curl https://sso-broker.production.univention-id-broker.com/auth/realms/ID-
↪Broker/broker/$NAMESPACEID/endpoint/descriptor > metadata.xml
```

4. Finally, you must introduce the ID Broker's service **Keycloak** as a service provider to the local IDP. Run the following command to save the appropriate configuration in the IDM of the *school authority*:

```
$ udm saml/serviceprovider create \
> --position "cn=saml-serviceprovider,cn=univention,$(ucr get ldap/base)" \
> --set serviceProviderMetadata="$(cat metadata.xml)" \
> --set AssertionConsumerService="https://sso-broker.production.univention-id-
↪broker.com/auth/realms/ID-Broker.com/broker/$NAMESPACEID/endpoint" \
> --set Identifier="https://sso-broker.production.univention-id-broker.com/
↪auth/realms/ID-Broker/broker/$NAMESPACEID/endpoint/descriptor" \
> --set isActivated=TRUE \
> --set simplesamlNameIDAttribute=entryUUID \
> --set simplesamlAttributes=TRUE \
> --set attributesNameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"␣
↪\
> --set LDAPattributes=entryUUID
```

CHAPTER

FIVE

ERROR HANDLING

If a problem prevents the synchronization of a user, group or school to the ID Broker, the ID Connector will move the queue item into the directory `/var/lib/univention-appcenter/apps/ucsschool-id-connector/data/out_queues/$NAMESPACEID/trash`. This allows an administrator to inspect the content of the file and react appropriately.

After handling a file in the *trash* directory, it should be deleted.

It is possible to move the file back into the out-queue directory `/var/lib/univention-appcenter/apps/ucsschool-id-connector/data/out_queues/$NAMESPACEID` and let the ID Connector pick it up. The file may contain out of date LDAP data, which can inadvertently make the situation worse by for example reverting successful group membership changes. We recommend to use the script mentioned in *Synchronize a single user* (page 15) to generate a queue item with fresh LDAP data.

5.1 Synchronize a single user

The script `schedule_user` accepts a single username as command line argument. It triggers the ID Connector listener module to generate an in-queue item with fresh LDAP data.

```
$ univention-app shell ucsschool-id-connector /ucsschool-id-connector/src/schedule_
↪user demo_student
```

5.2 Synchronize a single school group

If a school class is missing, then `schedule_user <member>` can be used with one of the groups members, to trigger its synchronization. If a school group exists and should be updated, then the `schedule_group` command can be used.

```
$ univention-app shell ucsschool-id-connector /ucsschool-id-connector/src/schedule_
↪group demo_class
```

5.3 Synchronize a school

The script `schedule_school` accepts a school name and a number of parallel tasks as command line argument. It triggers the ID Connector listener module to generate an in-queue item with fresh LDAP data.

```
$ univention-app shell ucsschool-id-connector /ucsschool-id-connector/src/schedule_
↪school demoschool 12
```

5.4 Reinitialize synchronization of all users and groups

You can reinitialize the synchronization of *all* user and group objects with the following command:

```
$ univention-directory-listener-ctrl resync ucsschool-id-connector
```

Note: A reinitialization requires roughly the same synchronization time as the initial synchronization.

5.5 Error handling for `manage_schools_to_sync.py`

The script will exit if the school was already added to the school authority configuration. By passing the flag `--force` you can overwrite this behavior in case you want to synchronize the school again.

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py add_schools --school_authority Traeger2 --force ou1␣
↪ou2
```

The script can also add all existing schools to the school authority configuration without running the initial synchronization step which synchronizes all users and group objects.

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py add_schools --school_authority Traeger2 --all_
↪schools --initial_sync false
```

Instead of running the previous command for all existing schools, it can also be called for a specific school `ou1`.

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py add_schools --school_authority Traeger2 --initial_
↪sync false ou1
```

If the schools have already been removed from the school authority configuration but it is uncertain if they were removed from the ID Broker, the script can also be called with `--force`.

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py remove_schools --school_authority Traeger2 --force␣
↪ou1 ou2
```

Schools can also only be removed from the school authority configuration and not deleted from the ID Broker by passing the `--delete_schools false` when removing the school:

```
$ univention-app shell ucsschool-id-connector
$ cd /var/lib/univention-appcenter/apps/ucsschool-id-connector/conf/plugins/
↪packages/idbroker/
$ ./manage_schools_to_sync.py remove_schools --school_authority Traeger2 --delete_
↪schools false ou1 ou2
```

CHAPTER

SIX

USAGE OF SERVICES

If you successfully completed the steps described in the sections *Installation* (page 9) and *Configuration* (page 11), all school users and school groups of your UCS@school system will be synchronized to the ID Broker system. These users will then be able to log in through SSO using the ID Broker.

All users can use all services registered to the ID Broker. Services, which use the *Self-disclosure API*, can retrieve pseudonymized user information like roles and group membership.

CHAPTER

SEVEN

GLOSSARY

Identity Provider (IDP)
Instance that provides information to authenticate and authorize identities. In case of ID Broker scenarios this is typically a SAML or OpenID Connect IDP hosted by a *School Authority*.

Provisioning API
REST API of the ID Broker. *School authorities* use the API to send pseudonyms and a limited set of meta information on users and groups to the ID Broker.

School Authority
In context of this document, the term *school authority* subsumes various institutions which serve one or several schools with IT infrastructure. The school authority is the data source for all students and teachers of an environment. The ID Broker will receive a minimal subset of this data, see *Big Picture of Univention ID Broker* (page 3). This can be a single school, a school authority with several schools, or an environment hosting services for a federal state. The environments are hosting a UCS@school domain.

Service
In the context of this document a *service* is an application, which uses single sign-on with the ID Broker and provides a service for students and teachers. For example a learning platform, that offers books.

Service Provider (SP)
Instance that provides a *service*.

Self-disclosure API
REST API of the ID Broker which allows retrieval of meta information of an authorized user. It focuses on the role of the user and the assigned learning groups.

CHAPTER

EIGHT

CHANGELOG

This changelog lists major and minor version changes of the debian package **id-broker-id-connector-plugin**.

v1.3.18 (2024-03-07)

- Schools can be configured manually to be connected to the ID Broker by using the script **manage_schools_to_sync.py**. Read more in *UCS@school ID Connector Plugin configuration* (page 11). **Requirement**: The version of the **UCS@school ID Connector** must be 2.3.3 or later:

v1.2.26 (2024-01-30)

- A bug was fixed: School classes are now deleted as expected when they are deleted on the school authority side.

v1.2.24 (2024-01-08)

- Support to synchronize workgroups to the ID Broker. Read more in *UCS@school ID Connector Plugin configuration* (page 11).

v1.0.0 (2022-06-22)

- Synchronize school users, school classes and schools to the ID Broker.
- Implementation of the initial import mode which allows to onboard quickly to the ID Broker.

INDEX