Authentication

security, authentication, authenticator

Basic view

From the simple viewpoint, authentication mechanism in CzechIdM can be divided into three areas:

- **First access**: Unauthenticated users come to CzechIdM for the first time. They fill in their login and password and log in.
- **Following requests**: Users make requests for various pages. Every request needs to be authenticated again, for the backend to know who makes the request. There is no "session" held for the user on the server side. But of course users don't fill the credentials every time they make a request.
- **Single-sign-on (SSO)**: Unauthenticated users come to CzechIdM for the first time. Without the need to fill any login or password, they are authenticated to CzechIdM and come to the front page.

The **First access** part is simply a request to the `/authentication` endpoint, which is handled by LoginController. The authentication is then processed by configured authenticators (for more details see Authenticators). The result of the successful authentication is a **JWT token**, which is persisted (`IdmTokenDto`) and returned to the frontend client through HTTP headers.

Following requests need to contain the information about the user who makes the request. So the client appends the **JWT token** to the request `CIDMST` HTTP header or adds token to url `cidmst` parameter (both ways are supported). The `AuthenticationFilter` intercepts every request and lets all configured `IdmAuthenticationFilters` process the request. One of them is `JwtIdMAuthenticationFilter`, which load persisted token, verifies the JWT token, retrieves the user name from it and authenticates that user.

Of course, following requests can't use the same JWT token forever, because it contains also the expiration date of the token. Therefore, the ExtendExpirationFilter intercepts every requests. This filter verifies current JWT token and creates a **new one** with extended expiration date. Expiration is extended in one minute window. The frontend saves this new JWT token and uses it for following requests.

SSO is an additional ability of the filter mechanism, which can be enabled and extended. Since the filters intercept every request to a secured endpoint, they are also called when the endpoint `/authentication/remote-auth` is requested the first time the user comes to CzechIdM (it is requested, because the frontend is so implemented). So there can be some instance of `IdmAuthenticationFilter` which will e.g. check some cookie or HTTP header, send its value to some external authority and obtain corresponding user name. The filter then authenticates this user, creates the JWT token a returns it. In the end, the user is authenticated to CzechIdM, even if no credentials of the local CzechIdM user were given to the authentication endpoint. The example of such filter is `SsoIdmAuthenticationFilter`, which processes the username in the header `REMOTE_USER`.

Each authenticator can implement **logout** mechanism. Logout is called the same way as other requests - used authentication is resolved by registered `IdmAuthenticationFilter`, persisted token is loaded for the given request and then is available for authenticator's logout method. Logout mechanism is based on persisted token, because all authenticator implementations creates persisted token in the end (by `JwtAuthenticationService`). Created underlying token can be deleted / disabled - external token id can be used as token identifier in external system (has to be filled by authenticator, when token is created - use method `JwtAuthenticationService#createJwtAuthenticationAndAuthenticate` with prepared token).

Authenticators

Components implementing Authenticator are used as an authentication chain. Every such component checks authentication against some authority, see the actual list of authenticators.

Every authenticator defines how the result of its authentication should be processed. For possible types of results see (AuthenticationResponseEnum):

- **requisite** - when authentication fails, the authentication chain is immediately broken and throws an exception, the user won't be authenticated.
- **sufficient** - this authenticator is authoritative, i.e. when its authentication succeeds, the authentication chain **doesn't continue and is successful**.

When the authenticator throws an exception, the authentication chain may continue or be broken depending on its defined result type (see results' type). If the authenticator doesn't return any valid object of type LoginDto (i.e. return **NULL**), the authentication chain continues.

The authenticators are ordered. They are processed in defined order in the implementation of the class AuthenticationManager. The authentication manager also checks the module to which the authenticator belongs. Authenticators from inactive modules are removed from the authentication chain.

Actual list of authenticators

DefaultCoreAuthenticator

DefaultCoreAuthenticator is the standard authenticator of IdM. It checks LoginDto against user name and password in the local IdM repository.

The service LoginService is called in this authenticator.

The result type of this authenticator is **SUFFICIENT**, it's order is **0**.

Logout method is implemented - used persisted IdM token (`IdmToken`) is disabled.

DefaultAccAuthenticator

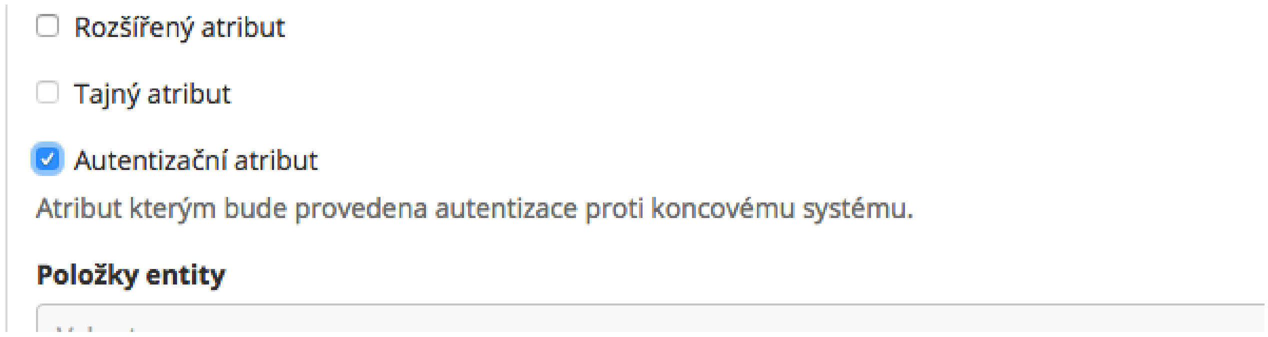
This authenticator is from version 10.4.0 deprecated. Please use DefaultAccMultipleSystemAuthenticator. The configuration properties will be removed.

DefaultAccAuthenticator checks the LoginDto against an end system. Users use their internal IdM login and their password from the end system.

The end system is defined in the configuration attribute (the UUID of the end system):

```
# ID system against which to authenticate
idm.sec.security.auth.systemId=
```

First, the authenticator tries to find an end system with given UUID. If such system doesn't exist, the authentication returns NULL. Otherwise, the authenticator finds the mapped attribute of the system which is marked as `Authentication attr.`.



- [ ] Rozšířený atribut
- [ ] Tajný atribut
- [x] Autentizační atribut

Atribut kterým bude provedena autentizace proti koncovému systému.

Položky entity

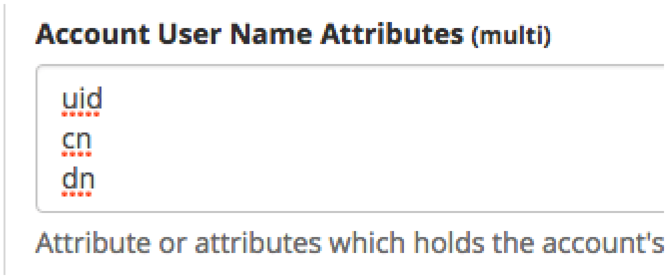
The authentication attribute is an attribute which represents the account on the system uniquely. This attribute will be used when calling the method **Authenticate** of the system connector. If there isn't any attribute marked as `Authentication attr.` in the system mapping, the identifier attribute is used by default.

Next, the authenticator finds the identity for given login. All accounts of this identity for the given system are processed and the method Authenticate is called for them one-by-one. The processing is stopped after the first successful authentication.

The result type of this authenticator is **SUFFICIENT**. It's order is **10**, which means that this authenticator would be processed after DefaultCoreAuthenticator

The authentication against an end system uses the system, which is defined as UUID in the configuration attribute **idm.sec.security.auth.systemId=**

If you want to authenticate against LDAP system, it's **necessary** to add the name of the authentication attribute (e.g. dn, or uid) to the field **Account User Name Attributes** in the system configuration, see the picture.


Account User Name Attributes (multi)

uid
cn
dn

Attribute or attributes which holds the account's

DefaultAccMultipleSystemAuthenticator

Since 10.4.0.

DefaultAccMultipleSystemAuthenticator has same behavior as DefaultAccAuthenticator but is allowed more system for authentication than one. Authenticator is placed between **DefaultAccAuthenticator** and **DefaultCoreAuthenticator**, more priority has original authenticator **DefaultAccAuthenticator**.

All behavior with authentication is same as original **DefaultAccAuthenticator**. The original authenticator is now deprecated. Please use the newer.

The end systems are defined with configuration properties. For example:

```
idm.sec.acc.security.auth.order1.system=e6a8b1e7-d656-47ae-aa2d-1062d1583c1a
idm.sec.acc.security.auth.order2.system=ea86a399-9b26-4f75-9b3a-d3f0049031ef
idm.sec.acc.security.auth.order3.system=
idm.sec.acc.security.auth.order4.system=
idm.sec.acc.security.auth.order5.system=SystemAD - User
idm.sec.acc.security.auth.order6.system=
idm.sec.acc.security.auth.order7.system=e6a8b1e7-d656-47ae-aa2d-1062d1583c1a
idm.sec.acc.security.auth.order8.system=
idm.sec.acc.security.auth.order9.system=LDAP User 2
```

Authentication is done from lowest order to highest. Maximum order is defined with the configuration property (default count is 50):

```
idm.sec.acc.security.auth.maximumSystemCount=50
```

Configuration property can be null or empty. **These configurations will be skipped**.