



## Authentication

Last updated: 03/03/2026

This content applies to the latest CD version of Cumulocity.

Specifications contained herein are subject to change and these changes will be reported in subsequent versions.

Copyright © 2026 Cumulocity GmbH.

The name Cumulocity GmbH and all Cumulocity GmbH product names are either trademarks or registered trademarks of Cumulocity GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

This software may include portions of third-party products. Third-party terms are set out in a 3rd-party-licenses file linked to or included with each installation package.

## Table of Contents

<b>Table of Contents</b>	<b>3</b>
<b>BASIC SETTINGS</b>	<b>4</b>
LOGIN SETTINGS	5
BASIC AUTH RESTRICTIONS	6
OAI-SECURE SESSION CONFIGURATION	6
OAI-Secure without the configuration related to the session management (session configuration turned off)	6
OAI-Secure with the configuration of the session management (session configuration turned on)	6
TOKEN GENERATION WITH OAI-SECURE	7
Lifespan configuration of JWT stored in the cookie	7
Lifespan configuration of cookies	7
Lifespan configuration of JWT in response body	7
TFA SETTINGS	8
<b>TWO-FACTOR AUTHENTICATION</b>	<b>9</b>
INTRODUCTION	9
SMS	9
TO ENABLE A SPECIFIC USER	9
TOTP	10
TO ENABLE A SPECIFIC USER	10
TO SET UP TOTP	10
TO REVOKE THE SECRET	12
TO DISABLE TOTP FOR A USER	13
<b>SINGLE SIGN-ON</b>	<b>14</b>
INTRODUCTION	14
CONFIGURATION SETTINGS	14
CONFIGURATION ACCESS	14
CONFIGURATION VIEW	15
CUSTOM TEMPLATE CONFIGURATION	15
TO CONFIGURE A CONNECTION	15
PLACEHOLDERS	21
CLIENT SUGGESTED IDENTITY PROVIDER	22
INTEGRATION WITH KEYCLOAK	23
GLOBAL LOGOUT FEATURE (AVAILABLE FOR KEYCLOAK IN VERSION 12.0.0 AND HIGHER)	23
LOGOUT ALL USERS FEATURE	23
INTEGRATION WITH AZURE AD	24
TO CONFIGURE AZURE AD	24
TO CONFIGURE SSO FOR AZURE AD IN CUMULOCITY	25
USING ACCESS TOKENS FROM THE AUTHORIZATION SERVER	28
TO CONFIGURE AUTHENTICATION WITH ACCESS TOKENS FROM AUTHORIZATION SERVERS	28
INTROSPECTION ENDPOINT	29
USER INFO ENDPOINT	30
TROUBLESHOOTING	31
INSPECT TOKEN CONTENT	31
ENFORCE THE USE OF THE TENANT DOMAIN IN SSO LOGIN	32

## BASIC SETTINGS

Click **Authentication** in the **Settings** menu if you want to view or change the basic authentication settings.

The screenshot shows the 'Authentication' settings page in the Cumulocity administration interface. The left sidebar lists various administration options, with 'Authentication' highlighted. The main panel is titled 'Authentication' and contains three main configuration sections. The 'Login settings' section allows configuring the preferred login mode (currently 'OAI-Secure'), password validity (0 days), password strength enforcement (enabled), and case sensitivity. The 'Basic Auth restrictions' section allows disabling basic authentication for web browsers and defining forbidden or trusted user agents. The 'OAI-Secure session configuration' section allows enabling session configuration and user agent validation, along with session timeout settings. A 'Save' button is located at the bottom left of the settings area.

## ✓ REQUIREMENTS

### ROLES & PERMISSIONS:

To see the **Authentication** menu item, you must have ADMIN permission for the “Tenant management” permission type or be the first admin user created in the tenant.

For easier user access management, the above permission(s) are/is included in the global role(s) created by default in every new tenant:

- Tenant manager - manages tenant-wide configurations like applications, tenant options and retention rules.

Additionally, access to the **Basic settings** tab may be restricted by the platform administrator via setting the `onlyManagementTenantAccess` option to `true` for Basic Auth or OAI-Secure login options (see [Cumulocity OpenAPI Specification](#)).

## i RELATED TOPICS

- [Platform administration > Authentication > Two-factor authentication](#) for details on the two-factor authentication strategies in Cumulocity.
- [Platform administration > Authentication > Configuring single sign-on](#) for details on configuring single sign-on in Cumulocity.
- [Authentication](#) in the Cumulocity OpenAPI Specification for details on managing authentication via

---

REST.

## LOGIN SETTINGS

In the **Preferred login mode** field, you can select one of the following options:

- OAI-Secure - recommended, since it provides high security, using authorization tokens to prove the identity of the user. Default login mode on creating new tenants. This mode is an enhancement of the previous OAuth Internal authentication (available prior to 10.13.0).
- Basic Auth - should be selected only for specific compatibility reasons, since it only provides basic security.
- Single sign-on redirect - can be selected only if SSO is configured. If selected, will remove Basic Auth and OAI-Secure login options.

This login mode will be used by the platform's applications as the default method to authenticate users. Device authentication stays unchanged.

### ! IMPORTANT

Each time you change the login mode you will be forced to log out. Other users will need to log out and log in again so that the change is applied.

In the field **Password validity limit**, you can limit the validity of user passwords by specifying the number of days after which users must change their passwords. If you do not want to force your users to change passwords, use "0" for unlimited validity of passwords (default value).

### i INFO

The password validity limit is not imposed on users with a "devices" role. This prevents device passwords from expiring.

By default, users can use any password with eight characters or more. If you select **Enforce that all password are "strong" (green)**, users must provide strong passwords as described in [To change your password](#).

### i INFO

The password validity limit and the password strength may not be editable, if configured by the platform administrator.

The **Ignore case when logging in** toggle allows enabling or disabling case sensitivity for the username or alias when authenticating a user login. If enabled, this feature is applied to all tenant users. By default, the feature is disabled.

### i INFO

The toggle can only be managed by a tenant administrator. Additionally, the feature can only be enabled if there are no case-insensitive collisions for the username or alias fields for all existing tenant users (excluding "device users"). The check for naming collisions is performed automatically when attempting to enable the feature.

## BASIC AUTH RESTRICTIONS

Even if OAI-Secure authentication is configured for users, basic authentication remains available for devices and microservices using the platform. To provide a higher security level the basic authentication can be restricted.

Use the **Forbidden for web browsers** toggle to disallow the usage of basic authentication for web browsers. Moreover you can specify the following parameters:

- **Trusted user agents** - this list is empty by default. If some user agent is added, all the HTTP requests containing this entry in the `User-Agent` header and having a valid basic authentication date will be accepted.
- **Forbidden user agents** - this list is empty by default. If some user agent is added, all the HTTP requests containing this entry in the `User-Agent` header and using basic authentication will be rejected.

### INFO

If the user agent is not found in the list of trusted or forbidden user agents then Cumulocity will try to verify if it is a web browser using an external library.

## OAI-SECURE SESSION CONFIGURATION

OAI-Secure is a more secure alternative to the Basic Auth mode that also supports username and password login. In OAI-Secure mode the credentials in the initial request are exchanged for a JWT token that is set as a cookie in the web browser or returned in the response body. Based on the configuration OAI-Secure can support full session management or work as a standard JWT authentication where the user session lifetime is limited by the token expiration time.

### OAI-Secure without the configuration related to the session management (session configuration turned off)

When there is no configuration related to the session, OAI-Secure issues a JWT token with a certain lifetime. If the token expires then the user is forced to re-login because token refresh is not supported. This behavior is very inconvenient for the user if the token lifetime is short because the user is forced to re-login frequently.

### OAI-Secure with the configuration of the session management (session configuration turned on)

Using OAI-Secure with session configuration is more convenient and secure, and can be used to achieve a behavior which is similar to the authentication based on HTTP sessions.

The OAI-Secure token acts as a session identifier on the client side (web browser). Such a token identifier which is stored in the cookie can have a preconfigured short lifetime. Then, the Cumulocity platform is responsible for renewing the session identifier without any user interaction. It is sufficient that the user's action causes the web browser to send a request to Cumulocity. Then, Cumulocity can examine if the renewing of the session identifier should be executed and perform the operation if necessary. Cumulocity offers extensive configuration related to this behavior so that tenant administrators can adjust the configuration to their needs.

If the **Use session configuration** option is enabled, the following settings can be configured on tenant level by a tenant administrator:

Field	Description	Default
User agent validation required	If turned on, the user agent sent in headers of consecutive requests in the scope of one session will be compared and a request with changed user agent will not be authorized.	false
Session absolute timeout	Defines the maximum period of time that the user can use Cumulocity without having to re-authenticate.	14 days
Session renewal timeout	Expected to be much shorter than the absolute timeout. Defines the time after which Cumulocity tries to provide a new token (session identifier). The renewal may take place only when Cumulocity receives an HTTP request from a client with a non-expired token and the period of time between obtaining the token and the execution of the request is greater than the renewal timeout.	1 day

Field	Description	Default
Maximum parallel sessions per user	Defines the maximum number of sessions which can be started by each user (for example on different machines or browsers). When a user exceeds this limit, then the oldest session will be terminated and the user will be logged out on this particular device.	5 sessions
Token lifespan	Defines the time for which a token is active. The user is only able to access Cumulocity with a valid token. This configuration option is always available, it does not depend on session configuration. See <a href="#">Token generation with OAI-Secure</a> below.	2 days

## INFO

The time parameters should depend on each other in the following manner: renewal timeout < token lifespan < absolute timeout. Moreover, the renewal timeout should be approximately half of the token lifespan.

Therefore, the recommended settings for a standard use case for OAI-Secure are the following:

- **Session absolute timeout:** 28 800 seconds (8 hours)
- **Session renewal timeout:** 2700 seconds (45 minutes)
- **Token lifespan:** 5400 seconds (90 minutes)

In such configurations, the idle timeout is in the range of 45 to 90 minutes, depending on when the last activity for the session was performed.

During the session token renewal the previous token is revoked and a new one is provided. The parameter `renewal token delay` defines the delay used to make this process smooth and not disturbing for the user. The old token is still valid for this period (1 minute by default). This way both tokens, old and new, are accepted by Cumulocity. This parameter is only configurable on platform level and cannot be modified by the tenant administrator.

## TOKEN GENERATION WITH OAI-SECURE

OAI-Secure is primarily based on JWT stored in a browser cookie. It can be also used to generate JWT in the response body. The lifespan of the tokens and the cookie is configurable by tenant options belonging to the category `oauth.internal`.

### Lifespan configuration of JWT stored in the cookie

JWT tokens stored in the browser cookie have a default validity time of two weeks. This can be changed with tenant options:

- category: `oauth.internal` ;
- key: `basic-token.lifespan.seconds` ;

The minimum allowed value is 5 minutes.

### Lifespan configuration of cookies

Cookies used to store a JWT token in a browser have their own validity time that can be changed with tenant options:

- category: `oauth.internal` ;
- key: `basic-user.cookie.lifespan.seconds` ;

The default value is two weeks. To have the cookie deleted when the user closes the browser, set it to any negative value.

### Lifespan configuration of JWT in response body

The lifespan of JWT tokens generated in the response body is configured with the following tenant options:

- category: `oauth.internal` ;
- key: `body-token.lifespan.seconds` ;

Refer to the [Tenant API](#) in the Cumulocity OpenAPI Specification for more details.

## INFO

If external communication to the Management tenant has been blocked, then it is only possible to access the tenant in a secure way (for example via SSH tunnel). This means that you can just as well use basic authentication. Additionally, it is not possible to use single sign-on since the communication from the external authorization server is also blocked. Therefore, the authentication method is automatically set to “Basic authentication” if the Management tenant is configured to block external communication.

## TFA SETTINGS

Select the checkbox **Allow two-factor authentication** if you want to allow TFA in your tenant (only possible for administrators).

You may select one of the following options:

- **SMS-based**, supporting the following settings:
  - **Token validity limit** - lifetime of each session in minutes. When the session expires or a user logs out, the user must enter a new verification code.
  - **Verification code validity limit** - here you can set the lifetime of each verification code sent via SMS. When the verification code expires, the user must request a new verification code in order to login.

Note that an SMS gateway microservice must be configured for the tenant. Naturally only users with a valid phone number associated can use this functionality.

- **TOTP** (Time-based One-Time Password) supporting the following setting:
  - **Enforce TOTP two-factor authentication on all users** - when enabled it will force all users to set up their TFA on login. Otherwise each individual user can choose to activate it or not.

Note that the TOTP method is only available with the login mode “OAI-Secure”.

Click **Save TFA settings** to apply your settings.

## IMPORTANT

- Each time you change the TFA method you will be forced to log out. User TFA settings are cleared and must be configured again.
- Users with a “devices” role are excluded from TFA and TOTP. This is also true when TOTP is enforced for all users.



## TWO-FACTOR AUTHENTICATION

### INTRODUCTION

The two-factor authentication (TFA) is an extra layer of security that only completes authentication with a combination of two different factors: something the users know (username and password) and something they have (for example, smartphone) or something they are (for example, fingerprint).

There are two possible TFA strategies: Short Message Service (SMS) and Time-based One-Time Password (TOTP). Only one of them can be active at a time.

To check whether TFA is enabled for a certain user, go to the **Users** page and see the TFA status column right from the password strength column. A key icon indicates that TFA is enabled and by hovering over it you can see the strategy that is being used.



### **i** RELATED TOPICS

- [Platform administration > Authentication > Basic settings](#) for information on how to configure basic authentication settings.
- [Authentication](#) in the Cumulocity OpenAPI Specification for details on managing authentication via REST.

## SMS

### **✓** REQUIREMENTS

When adding a user and TFA is enabled, a mobile phone number must be specified. Without a valid phone number a login is impossible.

### TO ENABLE A SPECIFIC USER

1. In the Administration application, navigate to **Accounts > Users** and select a user in the **Users** page.
2. Select the checkbox next to **Two-factor authentication (SMS)**.
3. Click **Save**.

**Login options**

- ☒ Two-factor authentication (SMS) ?
- ☐ User must change password on the next login

Cancel

Save

**i INFO**

This process can only be executed in the Administration application and is not available under **User settings**.

**TOTP****✓ REQUIREMENTS**

Users must install a TOTP application on their smartphone (Google Authenticator is recommended), freely available both on App Store and Play Store.

**TO ENABLE A SPECIFIC USER**

1. In the Administration application, navigate to **Accounts > Users** and select a user in the **Users** page.
2. Select the checkbox next to **Two-factor authentication (TOTP)**. This option is available only if the user has TOTP configured. If this is not the case, select **Enforce TOTP setup for the user**.
3. Click **Save**.

**Login options**

- ☐ Two-factor authentication (TOTP) ?
- ☐ Enforce TOTP setup for the user
- ☐ User must change password on the next login

Cancel

Save

**TO SET UP TOTP**

Opposed to the SMS strategy TOTP must be set up by each user. By opening **User settings** in the top right corner and then clicking **Set up two-factor authentication** they can start the setup process.

## Edit user

Email

john.doe@example.com

First name

John

Last name

Doe

Telephone

+48123456789

Product experience

☐ Enable personalized product experience tracking

Login options

Change password

Set up two-factor authentication

Cancel

Save

IF TFA is enabled, the user will be presented a QR code at login, that must be scanned with the previously installed TOTP mobile application.

Alternatively, the secret can also be inserted manually in case scanning the QR code is not an option.



Scan this QR code with your smartphone using the authenticator application.



IZTTFVKDIKNQH77VZELSA2T3KSPV6YJG

Verification code

e.g. 624327 (required)

 In case of key loss, please contact your platform administrator.

Cancel

Verify

After this process the mobile application will generate a new code every 30 seconds that can be used to complete the authentication process.

## TO REVOKE THE SECRET

If a user loses access to the TFA code, for example, if a user loses the phone or uninstalls the application, and needs to set it up again, the secret must be revoked. TOTP must be set up by each user individually.

## ✔ REQUIREMENTS

Users can not revoke their own TOTP secret. The secret of a user is only revoked by their respective parent user. See [Managing user hierarchies](#) for detailed information on user hierarchies.

ROLES & PERMISSIONS:

- To revoke a secret: ADMIN or CREATE permission for permission type "User management"

1. In the Administration application, navigate to **Accounts > Users** and select a user in the **Users** page.
2. Scroll down to **Login options**.
3. Click **Revoke TOTP secret**.
4. Confirm by clicking **Revoke**.

**Telephone**

e.g. +49 9 876 5

**Login options**

☒ Two-factor authentication

☒ Enforce TOTP

**Revoke TOTP secret**

This action will revoke the user's stored TOTP secret which will require a new setup.

## TO DISABLE TOTP FOR A USER

If a user wants to turn off the use of TOTP (and thus TFA) completely, the secret must be revoked and TOTP enforcement must be disabled. TOTP must be set up by each user individually.

## ✔ REQUIREMENTS

### ROLES & PERMISSIONS:

- To revoke a secret: ADMIN or CREATE permission for permission type "User management"
- To disable TOTP enforcement: ADMIN permission for permission type "User management"

To disable TOTP for a user follow these steps:

1. In the Administration application, navigate to **Accounts > Users** and select the user in the **Users** page.
2. Scroll down to **Login options**.
3. Click **Revoke TOTP secret**.
4. Confirm by clicking **Revoke**.
5. Clear the **Enforce TOTP setup for the user** checkbox.
6. Click **Save** to save your changes.

### Login options

- ☐ Two-factor authentication (TOTP) ?
- ☒ **Enforce TOTP setup for the user**
- ☐ User must change password on the next login

## SINGLE SIGN-ON

### INTRODUCTION

Cumulocity provides single sign-on (SSO) functionality, that allows a user to login with a single 3rd-party authorization server using the OAuth2 protocol, for example Azure Active Directory (AAD). Currently authorization code grant is supported with access and ID tokens in form of JWT. SAML is not supported. On top of standard SSO, Cumulocity also allows you to access the platform resources using access tokens from your authorization server directly as a Bearer token. For details refer to [Using access tokens from the authorization server](#).

### ✔ REQUIREMENTS

To use the SSO feature the following requirements must be met:

- The authorization server you use supports OAuth2 authorization code grant.
- The token is issued as JWT and you know what goes into the token content.
- The JWT must consist of a unique user identifier, “iss” (issuer), “aud” (audience) and “exp” (expiration time) fields.
- All microservices are built with Microservice Java SDK version 10.4.6 but preferably higher. For custom-built microservices, refer to [Security](#).
- For on premises installation the domain-based tenant resolution is configured properly.
- For Enterprise tenants, the enterprise domain must be set up as redirect URI in the basic configurations. If SSO providers have a list of allowed domains, the enterprise domain should be added to that list.
- You must assign a role to the user with at least READ permission for “Own user management”, otherwise the user cannot log in.
- Users must have cookies enabled in the browser settings, as the SSO feature is built on top of cookies technology.

### i RELATED TOPICS

- [Platform administration > Authentication > Basic settings](#) for information on how to configure basic authentication settings.
- [Authentication](#) in the Cumulocity OpenAPI Specification for details on managing authentication via REST.

### CONFIGURATION SETTINGS

To enable the SSO feature, the administrator must configure a connection with the authorization server. This is done in the Administration application.

### CONFIGURATION ACCESS

SSO configurations can be configured to be exclusively accessible by the Management tenant, thus preventing other tenants from accessing the configurations. Users of such tenants are unable to update the configuration. This removes the risk of an incorrectly configured SSO, which can prevent other users from logging in via SSO. The Management tenant can grant or restrict access to SSO configurations for specific tenants. For more information about configuration access, refer to the [Login options API](#) in the Cumulocity OpenAPI Specification.

## CONFIGURATION VIEW

Click the **Single sign-on** tab in the **Authentication** page. Note that the tab is only visible for tenants which have access to the SSO configuration.

At the top left, you can select a template. The selected option has an effect on the look of the panel. The default template is "Custom" which allows for a very detailed configuration with virtually any authorization server using OAuth2 authorization code grant. Other templates provide simplified views for well known and supported authorization servers. In the next steps there will first be a definition of how to use the "Custom" template followed by a view dedicated to Azure Active directory.

## CUSTOM TEMPLATE CONFIGURATION

On the **Single sign-on** tab in the **Authentication** page, select "custom" (the default) as template to configure a connection with any authorization server using OAuth2 authorization code grant.

As the OAuth protocol is based on the execution of HTTP requests and redirects, a generic request configuration is provided.

The screenshot displays the 'Single sign-on' configuration page. On the left is the 'ADMINISTRATION' sidebar with 'Authentication' selected. The main area shows the 'Single sign-on' configuration for the 'Custom' template. The 'Authorization request' URL is set to 'https://sso.io/auth/realms/t1/protocol/openid-connect/auth'. Below this is a 'Headers' section with an 'Add header' button. The 'Request parameters' section contains a table with the following data:

Key	Value
response_type	code
client_id	\${clientId}
redirect_uri	\${redirectUri}
scope	e.g. \${clientId} (required)

At the bottom of the configuration area is a 'Save' button.

## TO CONFIGURE A CONNECTION

The first part of the **Single sign-on** page consists of the request configuration. Here you can configure the HTTP request address, request parameters, headers and body in case of token and refresh requests. The authorize method is executed as a GET, token and refresh method by POST requests.



INFO

Be aware that the body field of each request, after filling placeholders with values, is sent in the request 'as is'. This means it is not encoded by Cumulocity. Many authorization servers require values inside the body to be URL-encoded (x-form-urlencoded). This can be achieved by entering already encoded values in a body field.

Specifying a logout request is optional. It performs a [front-channel single logout](#). If configured as a request parameter, you are redirected to the defined authorization server logout URL after logging out from Cumulocity. If you use the OpenID Connect standard for the integration with the authorization server, the ID token should be passed as the `id_token_hint` parameter in the logout request.

Logout request URL

`https://sso.io/auth/realms/t1/protocol/openid-connect/logout`

Request parameters


Key	Value
<code>id_token_hint</code>	<code>\${id_token}</code>
<code>redirect_uri</code>	<code>https://demo.cumulocity.com/</code>

[+ Add request parameter](#)

The **Basic** section of the **Single sign-on** page consists of the following configuration settings:

Basic [Redirect URL](#) [?](#)

`https://demo.cumulocity.com/tenant/oauth` ☐ Redirect to the user interface application [?](#)

 Make sure that "Valid Redirect URIs" in the authorization server is set to "<tenant\_domain>/tenant/oauth".

Client ID	Token issuer
<code>c8ydemo</code>	<code>https://sso.io/auth/realms/t1</code>
Button name	Provider name
<code>login with Keycloak and jwks</code>	<code>Keycloak</code>
Audience	
<code>account</code>	

☒ Visible on login page

Field	Description
Redirect URI	Redirect parameter. Can be used in request definitions as a <code>\${redirectUri}</code> placeholder. If <a href="#">Redirect to the user interface application</a> is enabled then the redirect URI is not required. The redirect URI will be set automatically to the currently used application.
Redirect to the user interface application	The redirect URL is automatically set to the application used by the user during login. If enabled remember to set valid redirect URIs in the authorization server, for example, " <code>https://cumulocity.com/apps/*</code> ". The advantage of enabling this option is that any errors that occur during the SSO configuration are displayed properly in the UI application.
Client ID	OAuth connection client ID. Can be used in request definitions as a <code>\${clientId}</code> placeholder



Field	Description
Token issuer	OAuth token issuer
Button name	Name displayed on the button on the <b>Login</b> page
Provider name	Name of the provider
Audience	Expected aud parameter of JWT
Visible on Login screen	Indicates whether the login option is enabled or not

Each time you log in, the content of the access token and ID token is verified and serves as the basis for your access to the Cumulocity platform. The following section provides the mapping between JWT claims and access to the platform.

Under **Source of dynamic access mapping**, the administrator can specify the source from which the JWT claims are retrieved, either access token or ID token.

#### Access mapping

Source of dynamic access mapping

☒ Retrieve from Access token  
☐ Retrieve from ID token

Dynamic access mapping principle

☐ Use dynamic access mapping only on user creation [?](#)  
☐ Roles selected in the rules below will be reassigned to a user on each log in and other ones will be unchanged [?](#)  
☒ Roles selected in the rules below will be reassigned to a user on each log in and other ones will be cleared [?](#)

Dynamic access mapping

No access mapping rules defined.  
Click below to add a new mapping.

+ Add access mapping

Inventory roles mapping

No inventory roles mapping rules defined.  
Click below to add a new mapping.

+ Add inventory roles mapping

Under **Dynamic access mapping principle** you can select one of the following options:

- **Use dynamic access mapping only on user creation**: When selected, dynamic access mapping will be used only when a new user logs in to fill in the initial roles. When a user already exists in Cumulocity, the roles will not be overwritten nor updated.
- **Roles selected in the rules above will be reassigned to a user on each log in and other ones will be unchanged**: When selected, dynamic access mapping is used on every login, but the roles not listed in the access mapping configuration are not updated. Only the global roles, default applications and device groups that are listed in the defined access mapping rules are overwritten.
- **Roles selected in the rules above will be reassigned to a user on each log in and other ones will be cleared**: This is the default. Dynamic access mapping assigns user roles, based on the token, on every user login. It is not possible to change the user roles inside Cumulocity as they would be overwritten on the next user login. To change this behavior, select one of the

remaining options.

If you select either of the first two options mentioned above, this will also allow administrators to edit the roles of SSO users in the user management. For details, refer to [Managing permissions and roles](#).

The dynamic access mapping configuration allows you to define the rules for assigning roles to users based on JWT claims. The rule that matches the token's value is used to assign the appropriate set of roles to the user.

Dynamic access mapping

When

Key	Operator	Value		
user	=	john.wick	-	and

Provide access to

Default global roles	Default applications
<div>X Business User</div>	<div>X cockpit</div>

+ Add access mapping

Inventory roles mapping

When

Key	Operator	Value		
user	=	john.wick	-	and

Provide inventory roles

Groups	Inventory roles
<div>Turbines (europe)</div>	<div><div>X Reader</div><div>X Asset Manager</div></div>

+ Add inventory roles


+ Add inventory roles mapping

In the example above, if a user tries to login a decoded JWT claims look like:

```
{
...
"user": "john.wick",
...
}
```

The user will be granted access to the global role “business”, the default application “cockpit” and the inventory roles “Manager” and “Reader” for the device group named “region north”.

If no access mapping matches the user token, you will get an “access denied” message when trying to log in. This will also happen if there is no access mapping defined causing all users to be unable to log in using SSO.



New rules can be added by clicking **Add access mapping** or **Add inventory roles** at the bottom. An access mapping statement can consist of multiple checks like in the image below. You can add a rule to an existing statement by clicking **and**. Click the remove icon  to remove a rule.

New roles are added to the user from every matching access mapping. If one access mapping statement assigns the role “admin” and a second one assigns the role “business” and both meet the defined conditions, then the user will be granted access to the global roles “business” and “admin”.

When using “=” as operator you may use wildcards in the **Value** field. The supported wildcard is asterisk (\*) and it matches zero or more characters. For example, if you enter “cur\*” this matches “cur”, “curiosity”, “cursor” and anything that starts with “cur”. “\*fn” matches “fn”, “fission”, “falcon”, and anything that begins with an “f” and ends with an “n”.

In case the asterisk character should be matched literally it must be escaped by adding a backslash (\). For example, to match exactly the string “Lorem\*ipsum” the value must be “Lorem\\*ipsum”.

When

Key	Operator	Value	
role	in	ADMIN	
user.type	=	human	 <span>and</span>

In this case the following claim will match the condition:

```
{
...
"user": {
  "type": "human"
},
"role": [
  "ADMIN"
],
...
}
```

On user login, user data like first name, last name, email and phone number can also be derived from JWT claims. Using the **Source of user data mapping** radio button, the administrator can decide whether the values should be retrieved from the access token or the ID token.

Based on that, the user data mappings can be configured:

## User data mappings

Source of user data mapping

☒ Retrieve from Access token  
☐ Retrieve from ID token

---

Claim names [?](#)

<b>First name</b> <input type="text" value="claim_name"/>	<b>Last name</b> <input type="text" value="claim_family_name"/>
<b>Email</b> <input type="text" value="claim_email"/>	<b>Phone number</b> <input type="text" value="claim_phone_number"/>

Each field represents the claim name that is used to retrieve the data from JWT. The user data mapping configuration is optional and as admin manager you can only use the required fields. If the configuration is empty or the claim name cannot be found in the JWT token then the values in the user data are set as empty.

Mapping for alias is not available because it is not used in the context of SSO login.

The username claim name can be configured in the **User ID** configuration window. The user ID can be set to any top-level field of the authorization token payload sent from the authorization server to the platform during the login process. We recommend you inspect the authorization token in the audit logs to make sure the correct field is used (see [Troubleshooting](#)).

If you check the **Use constant value** checkbox, a constant user ID is used for all users who log in to the Cumulocity platform via SSO. This means that all users who log in via SSO share the same user account in the Cumulocity platform. We do not recommend you to use this option.

Each token is signed by a signing certificate. The following options are available to configure the signing certificates.

1. By specifying the Azure AD certificate discovery address.

Signature verification

Verifier

Public key discovery URL

2. By specifying the ADFS manifest address (for ADFS 3.0).

Signature verification

Verifier

ADFS manifest URL

3. By providing the public key of a certificate manually to Cumulocity. A certificate definition requires an algorithm information, public key value and validity period.

Signature verification

Verifier  
Custom

Certificates

Type

☐ X.509 certificate (PEM format)

☒ RSA public key (X.509 Subject Public Key Info)

Public key in PEM format  
(required)

Valid from  
Date from (required)

Valid till  
Date to (required)

+ Add certificate

4. By specifying the JWKS (JSON Web Key Set) URI. JWKS is a set of JWK objects containing a public key used to verify tokens issued by the authorization server.

Signature verification

Verifier  
JWKS

JWKS URL  
<https://jwks.tenant.com/FederationMetadata/2007-06/FederationMetadata.xml>

## INFO

Cumulocity only supports certificates with RSA key, either as a ("n", "e") parameters pair or "x5c" certificate chain. Other key types (for example Elliptic-curves) are not supported.

## PLACEHOLDERS

Inside some fields you can use placeholders that are resolved by Cumulocity at runtime. Available placeholders are:

Placeholder	Description
clientId	Value of the <b>Client ID</b> field
redirectUri	Value of the <b>Redirect URI</b> field
code	Code returned by the authorization server in response to authorization request
refreshToken	Refresh token returned by the authorization server after token request
id_token	A JWT token issued by the authorization server that contains claims about the authenticated user; can be used for logout from the authorization server
idp_hint	Value of the <b>idp_hint</b> parameter passed by the UI application, to select which IdP should be preselected during the authentication flow

These placeholders can be used in authorization requests, token requests, refresh requests and logout request in the fields:

- URL
- Body
- Headers
- Request parameters

To use a placeholder in a field, put it inside two curly brackets preceded with a dollar sign:

Request parameters

Key	Value	
response_type	code	⊖
client_id	\${clientId}	⊖
redirect_uri	\${redirectUri}	⊖
scope	e.g. \${clientId} (required)	⊖

[+ Add request parameter](#)

Placeholders can also be used as a part of text:

Body

```
grant_type=authorization_code&code=${code}&redirect_uri=${redirectUri}&client_id=${clientId}
```

## INFO

Placeholders are not validated for correctness. Any not recognized or misspelled placeholder will be left in text unprocessed.

## CLIENT SUGGESTED IDENTITY PROVIDER

When the default login mode is **Single sign-on redirect**, and the authentication server supports multiple identity providers, instead of requesting the user to select an identity provider, the `idp_hint` parameter can be collected by Cumulocity and forwarded to the authentication server as part of the initial authorization request. This ensures that the user is redirected directly to the specified IdP.

The `idp_hint` is a custom parameter supported by the application to improve the login experience in multi-identity provider (IdP) scenarios. It allows a client application (for example, a custom welcome page) to suggest which IdP should be preselected during the authentication flow.

1. A client custom application appends the `idp_hint` parameter to the login URL:

```
https://cumulocity.com/apps/public/login/index.html?idp_hint=google
```

2. The Cumulocity application extracts the value of `idp_hint` (for example, "google").
3. During the OAuth2/OIDC authorization request, this value is mapped and passed to the underlying authentication server.

### Example with Keycloak

Keycloak supports an [identity provider hinting feature](#) using the `kc_idp_hint` parameter.

If the `idp_hint` value is "google", and the authorization request headers are configured with key `kc_idp_hint`, value `${idp_hint}`, Cumulocity will translate it into a Keycloak compatible parameter and forward it:

```
https://keycloak.com/realms/myrealm/protocol/openid-connect/auth
?kc_idp_hint=google
&...
```

In this example:

- A custom "welcome page" provides `idp_hint=google`.
- Cumulocity translates this hint into Keycloak's `kc_idp_hint=google`.
- The user is immediately redirected to the Google IdP without having to manually select it from the Keycloak login screen.

## INTEGRATION WITH KEYCLOAK

### GLOBAL LOGOUT FEATURE (AVAILABLE FOR KEYCLOAK IN VERSION 12.0.0 AND HIGHER)

Integration with Keycloak allows administrators to use a global logout feature based on OpenId Connect. An event from the Keycloak authorization server is sent to all applications (including the Cumulocity platform) with a logout token that is verified in the same way as the token used in the login process. This feature allows ending sessions on both sides, applications and Keycloak, for the particular user.

To configure the global logout feature follow these steps:

1. Go to the administrator console.
2. Select the realm used in the SSO configuration for the tenant.
3. Navigate to **Clients** in the **Configure** section.
4. Select the client used in the SSO configuration.
5. Set the **Backchannel Logout URL** field to "https://mytenant.cumulocity.com/user/logout/oidc".

To use the global logout feature follow these steps:

1. Go to the administrator console.
2. Select the realm used in the SSO configuration for the tenant.
3. Navigate to **Users** in the **Manage** section.
4. Select the particular user.
5. Navigate to the **Sessions** tab in the **Manage** section and click **Logout**.

### LOGOUT ALL USERS FEATURE

Keycloak also provides a feature which allows administrators to logout all SSO users.

To configure the logout all users feature follow these steps:

1. Go to the administrator console.
2. Select the realm used in the SSO configuration for the tenant.
3. Navigate to **Clients** in the **Configure** section.
4. Select the client used in the SSO configuration.
5. Set the **Admin URL** to "https://mytenant.cumulocity.com/user/keycloak"

To use the logout all users feature follow these steps:

1. Go to the administrator console.
2. Select the realm used in the SSO configuration for the tenant.
3. Navigate to the **Sessions** tab in the **Manage** section and click **Logout all**.

Note that the logout event for all users is only performed in the scope of one Keycloak realm. Moreover, it is only sent for those tenants where the client being used as a configuration for the SSO feature has the correct **Admin URL** value.

In the **Session** tab, the Keycloak administrator can also check how many active sessions exist on the respective client and estimate how many tenants and users will be affected by the logout event.

To confirm if the logout event for all users or a single user has been received by the tenant, the Cumulocity administrator can verify if there is information about the logout event in the audit logs. The audit logs are available in the Administration application under **Accounts** in the **Audit Logs** tab.

## INTEGRATION WITH AZURE AD

The integration was successfully verified against Azure AD using OAuth2 and OpenID Connect (SAML is not supported). The configuration steps are available in <https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-protocols-oauth-code>.

The following steps illustrate how to use Azure AD (Azure Active Directory) for SSO in Cumulocity.

### ✓ REQUIREMENTS

You need administrative access to your Azure AD.

### TO CONFIGURE AZURE AD

To connect Cumulocity to Azure AD, you must create an App registration in Azure AD.

1. Select **App Registrations** under **Manage** on the left and at the top click **New Registration**.
2. In the resulting window, provide a name for the new App registration.
3. As **Redirect URI type** select "Web" and enter the URL to your tenant OAuth endpoint, for example "https://documentation.cumulocity.com/tenant/oauth". You can derive this value from your Cumulocity tenant. Navigate to **Administration > Settings > Authentication > Single sign-on**. The redirect URL is prefilled by the platform. Optionally, set for example, "https://<tenant\_domain>/apps/\*" if in the Cumulocity SSO configuration **Redirect to the user interface application** is enabled.
4. Click **Register** to create the App registration.

The overview in the details page of your App registration contains several IDs and endpoints that you need later on, like the Application (client) ID and the Directory (tenant) ID (for your tenant in Cumulocity).

 Delete  Endpoints  Preview features

#### ^ Essentials

Display name	: <a href="#">CumulocityEdge</a>	Client credentials	: <a href="#">0 certificate, 1 secret</a>
Application (client) ID	: 7fd1ed48-f4b6-4362-b0af-2b753bb1af2b	Redirect URIs	: <a href="#">1 web, 0 spa, 0 public client</a>
Object ID	: 65dc57a6-a355-4ca4-ba7d-19208c838eb1	Application ID URI	: <a href="#">Add an Application ID URI</a>
Directory (tenant) ID	: 4d17551b-e234-4e18-9593-3fe717102dfa	Managed application in l...	: <a href="#">CumulocityEdge</a>
Supported account types	: <a href="#">My organization only</a>		

Moreover, the App registration requires a secret which is used by Cumulocity for the authentication.

1. In the details page of your App registration, click **Certificates & secrets** under **Manage** on the left.
2. Select **New client secret**.
3. Enter a description and select an expiry time.
4. Click **Add** to add the secret.

### ⚠ CAUTION



- Copy the value of the new secret to another location. It will no longer be visible once you have left the page.
- The secret string must not include a "=" character as this may conflict with the later usage in a URL. If it does, create a new one.

Optionally, create a user in Azure AD that you would like to use with Cumulocity.

## TO CONFIGURE SSO FOR AZURE AD IN CUMULOCITY

Navigate to **Settings > Authentication** in the Administration application and switch to the **Single sign-on** tab.

Retrieve the relevant information by a GET request to:

[https://login.microsoftonline.com/&lt;Directory tenant ID>/well-known/openid-configuration](https://login.microsoftonline.com/&lt;Directory%20tenant%20ID>/well-known/openid-configuration)

The response will look like this:

```

{
  "token_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/oauth2/token",
  "token_endpoint_auth_methods_supported": [
    "client_secret_post",
    "private_key_jwt",
    "client_secret_basic"
  ],
  "jwks_uri": "https://login.microsoftonline.com/common/discovery/keys",
  "response_modes_supported": [
    "query",
    "fragment",
    "form_post"
  ],
  "subject_types_supported": [
    "pairwise"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "response_types_supported": [
    "code",
    "id_token",
    "code id_token",
    "token id_token",
    "token"
  ],
  "scopes_supported": [
    "openid"
  ],
  "issuer": "https://sts.windows.net/4d17551b-e234-4e18-9593-3fe717102dfa/",
  "microsoft_multi_refresh_token": true,
  "authorization_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/oauth2/authorize",
  "device_authorization_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/oauth2/devicecode",
  "http_logout_supported": true,
  "frontchannel_logout_supported": true,
  "end_session_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/oauth2/logout",
  "claims_supported": [
    "sub",
    "iss",
    "cloud_instance_name",
    "cloud_instance_host_name",
    "cloud_graph_host_name",
    "msgraph_host",
    "aud",
    "exp",
    "iat",
    "auth_time",
    "acr",
    "amr",
    "nonce",
    "email",
    "given_name",
    "family_name",
    "nickname"
  ],
  "check_session_iframe": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/oauth2/checksession",
  "userinfo_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/openid/userinfo",
  "kerberos_endpoint": "https://login.microsoftonline.com/4d17551b-e234-4e18-9593-3fe717102dfa/kerberos",
  "tenant_region_scope": "EU",
  "cloud_instance_name": "microsoftonline.com",
  "cloud_graph_host_name": "graph.windows.net",
  "msgraph_host": "graph.microsoft.com",
  "rbac_url": "https://pas.windows.net"
}

```

Now enter the following values in the configuration:

Azure	Cumulocity	Value
Login URL; OpenID config; Beginning of token endpoint	Azure AD address	Address of your Azure AD tenant, for example "https://login.microsoftonline.com"
Home > Overview > Primary Domain	Tenant	<directoryName>.onmicrosoft.com, for example "admtest.onmicrosoft.com"
OpenID config "issuer"	Token issuer	Token issuer value in form of a HTTP address: "https://sts.windows.net/<Directory tenant ID>/" . Note that this won't work without the trailing slash.
App registration > <app> > Application (client) ID	Application ID	for example "7fd1ed48-f4b6-4362-b0af-2b753bb1af2b"
Redirect URI		Address of your Cumulocity tenant followed by "/tenant/oauth", or optionally followed by "/apps/*" when <b>Redirect to the user interface application</b> is enabled.
App registration - <app> > Certificates & secrets > Value	Client secret	Azure AD client secret, for example "hE68Q~uC1.BISzGJSDC3_UEFvvyIZvRcCxbvV345"
From OpenID config	Public key discovery URL	"https://login.microsoftonline.com/common/discovery/keys" or "https://login.microsoftonline.com//discovery/keys"

Optionally single logout can be configured:

Field	Description
Redirect after logout	Activates single logout by redirecting the user, after logout, to the authorization server logout endpoint
Redirect URL	Address to redirect the user to after successful logout from the authorization server

After configuring SSO in Cumulocity, you can try to login. You might get an "access denied" error, if this user has no access mapping yet. But you should see a "User login" event and a JSON web token in the audit logs (**Administration > Accounts > Audit logs**).

The content looks like this:

```
{
  "typ": "JWT",
  "alg": "RS256",
  "x5t": "2ZQpJ3UpbjAYXYGaXEJl8lV0TOI",
  "kid": "2ZQpJ3UpbjAYXYGaXEJl8lV0TOI"
}{
  "aud": "7fd1ed48-f4b6-4362-b0af-2b753bb1af2b",
  "iss": "https://sts.windows.net/4d17551b-e234-4e18-9593-3fe717102dfa",
  "iat": 1660815959,
  "nbf": 1660815959,
  "exp": 1660820080,
  "acr": "1",
  "aio": "ASQA2/8TAAAg0xPUeu6HKAlgK3vZJsW8TdejNB3BGsz4XFmJLzPt0=",
  "amr": [
    "pwd"
  ],
  "appid": "7fd1ed48-f4b6-4362-b0af-2b753bb1af2b",
  "appidacr": "1",
  "family_name": "Doe",
  "given_name": "Jane",
  "ipaddr": "51.116.186.93",
  "name": "Jane Doe",
  "oid": "afbff765-592e-4ae1-9334-b968dad59c84",
  "rh": "0.AXkAG1UXTTTtGE6Vgz_nFxAt-kjt0X-29GJDsK8rdTuxryuUAAw.",
  "scp": "openid User.Read User.Read.All User.ReadBasic.All",
  "sub": "zRTnTukAjU11ME1aqiPMOdwk9jVNmInXbeuoUr_3cYk",
  "tid": "4d17551b-e234-4e18-9593-3fe717102dfa",
  "unique_name": "jane@admtest.onmicrosoft.com",
  "upn": "jane@admtest.onmicrosoft.com",
  "uti": "lcTqpKPIA0G_P1Lyw6xBAA",
  "ver": "1.0"
}[
  256 crypto bytes
]
```

You can now use the claims to map user attributes and give permissions in the same way as described in [Custom template configuration](#).

## USING ACCESS TOKENS FROM THE AUTHORIZATION SERVER

You can directly request Cumulocity to use OAuth2 access tokens from your authorization server. This way, your applications or users can access resources without logging in to the platform or using Basic authentication. This leverages your authorization server to get access tokens for your applications which you can send in subsequent request to Cumulocity.

### ✔ REQUIREMENTS

This feature requires the following on top of the general requirements:

- Your authorization server must support the OAuth2 client credentials grant type.
- All microservices are built with Microservice Java SDK version 1018.6.0 or higher. For custom-built microservices, refer to [Security](#).

## TO CONFIGURE AUTHENTICATION WITH ACCESS TOKENS FROM AUTHORIZATION SERVERS

Enable or disable this authentication option in the **External token configuration** section.

External token configuration ☐ Allow authentication with access token from external IAM system

If enabled, this authentication takes precedence over the standard [JWT token authentication](#), which means, for example, that an HTTP request to Cumulocity with the header `Authorization: Bearer {{access token}}` assumes that the source of the access token is your authorization server instead of the token being issued by Cumulocity. Configure the user ID or the application ID to any top-level claim in the access token.

External token configuration ☒ Allow authentication with access token from external IAM system

User/App ID

JWT field

appld

☐ Use constant value

☐ Validate access token

Cumulocity creates a user which gets assigned the configured user ID or application ID. Additionally, this user is granted the roles to access to the applications defined in the **Access mapping** section.

## INFO

If it is set, the configuration allows you to create a Cumulocity user representing your applications (the access tokens are obtained via the *client credentials flow*), or the users of your authorization server (the access tokens are obtained with the *password grant type*).

By default, Cumulocity verifies that the token is not expired and its signature matches the signature you have configured earlier. You can strengthen the validation of the token by configuring either an introspection or a user info validation with the necessary credentials. This way, the platform knows if the access tokens were intentionally invalidated or expired. You cannot access Cumulocity resources with an invalidated access token.

## INTROSPECTION ENDPOINT

Cumulocity uses token introspection to verify the validity of the access tokens of your applications. In general, this endpoint can be used for access tokens obtained via the client credentials flow or any other OAuth2 flow.

To configure the introspection, provide an introspection endpoint and a URL-encoded (x-form-urlencoded) body containing the access token, the client ID and the client secret, and an "Authorization" request header.

Cumulocity requests the introspection endpoint of your authorization server to query the status of the access token. If the token is active, proceed with verifying the token signature.

## External token configuration

☒ Allow authentication with access token from external IAM system

User/App ID

JWT field

appld

☐ Use constant value☒ Validate access token

Validation method

Introspection

Token validation request

URL

https://sso.c8y.io/auth/realms/staging-latest-jwks/protocol/openid-connect/token/introspect

Body

token=\${accessToken}&amp;client\_id=\${clientId}&amp;client\_secret=

Headers

+ Add header

Request parameters

+ Add request parameter

Access token validation frequency

1

minutes

You can configure the **Access token validation frequency** to set how often the introspection is performed as it may be costly to always call the authorization server for the same access token. The validation status of the token is cached internally for the specified time. If the token is revoked in the meantime, Cumulocity will only be aware during the next validation, that is, the token is still considered until the next validation. To avoid this, use a frequency. The default value is one minute.

## Access token validation frequency

1

minutes

## USER INFO ENDPOINT

The user info request can also be used to check the validity of the access token of your users. Unlike introspection, a user info request requires a user context. This means you cannot use it to validate access tokens obtained with the client credentials flow.

External token configuration

☒ Allow authentication with access token from external IAM system

User/App ID

JWT field

userid ☐ Use constant value

☒ Validate access token

Validation method

Introspection

Token validation request

URL

https://sso.c8y.io/auth/realms/staging-latest-jwks/protocol/openid-connect/userinfo

Body

token=\${accessToken}&client\_id=\${clientId}&client\_secret=

Headers

+ Add header

Request parameters

+ Add request parameter

Access token validation frequency

1 minutes

## ⚠ CAUTION

If you use one of the two validation methods, make sure that your authorization server exposes the introspection or the user info endpoint.

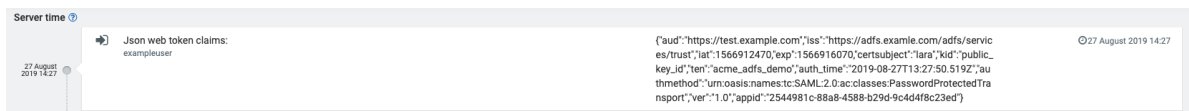
## TROUBLESHOOTING

### INSPECT TOKEN CONTENT

It can be particularly helpful to inspect the content of the authorization token sent to the platform as some of its fields contain the information required for the correct configuration described above.

In the Administration application, click **Accounts > Audit logs**, filter by the type “Single sign-on” and look for entries with “JSON web token claims”.

The contexts of the token will be presented in JSON format.



## ENFORCE THE USE OF THE TENANT DOMAIN IN SSO LOGIN

If a tenant is configured to use the `tenantId` in the `baseUrl` (instead of a tenant domain), users may experience unexpected redirect behavior after authenticating via SSO.

If a user opens an application by entering its URL directly and initiates login via SSO, they may not be returned to the intended application after authentication. Instead, they may be redirected to the default application configured for the tenant.

This can be changed by setting the following tenant options:

- category: `sso`
- key: `sso-redirect-default-application`
- value: `false`

Setting this tenant option to `false` will enforce the use of the tenant domain during SSO login. This results in:

- Correctly scoped cookies being set for the intended application, ensuring the user is redirected back to the original application after successful authentication.
- Support for SSL certificates using **Subject Alternative Name (SAN)**, eliminating the need for wildcard or tenant-specific certificates.

### When to use:

Apply this setting when:

- The tenant environment is configured with base URLs that include `{tenantId}` instead of `{tenantDomain}`.
- Users are redirected to the wrong application after SSO login.
- You want to use a single SSL certificate with SAN entries instead of maintaining separate certificates for each domain.