

KNIME Secrets User Guide for KNIME Community Hub

KNIME AG, Zurich, Switzerland
Version 1.14 (last updated on)



Table of Contents

Introduction.....	1
Managing secrets.....	2
Create a new secret.....	2
Edit an existing secret.....	3
Delete a secret.....	3
Interactive login.....	3
Manage access to team secrets.....	4
Using secrets.....	5
Secret types.....	6
Choosing the right secret type.....	6
Box.....	7
Credentials.....	9
Databricks.....	10
File.....	13
Generic OAuth2.....	14
Google.....	16
Microsoft.....	19
Salesforce.....	24
How-tos.....	26
Create Azure App for user authentication.....	26
Create Azure App for application/service principals.....	30
Create Azure Storage SAS URL.....	31
Find the Azure Storage shared key.....	32
Architecture.....	33

Introduction



KNIME Secrets on KNIME Community Hub are available only for **Team plan** users.

Secrets provide a way to centrally store and manage logins to other systems. For example, a secret could be credentials to log into an external database, file system or service. Secrets are owned and managed by a user or a team admin. User secrets are intended for managing personal logins e.g. john.smith. Team secrets on the other hand are intended for shared logins sometimes referred to as technical or service users, e.g. hr_read_only, that are shared with multiple users.

Managing secrets

Secrets are managed via the KNIME Community Hub.

- To manage your personal secrets navigate to your account page and select *Secrets* from the menu on the left. On your *Secrets* page you can create, edit and delete your personal secrets.
- To manage team secrets you have to navigate to the team page you want to manage the secrets for. Once you are on the team page select *Secrets* from the menu on the left.

Creation, editing and deletion for personal, team and admin secrets works the same and is described below. Team secrets can only be created by team admins.

KNIME Dev Business Hub > Documentation team > Secrets

Secrets of Documentation team

DT VM Manage team

Rows: 4 All time Grouped by 'Owner' Q

Name	Owner ↓	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is company wide used login for the...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared secret for HR database
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-only user for the HR DB
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login for development HR

Rows: 4 25 per page

Create a new secret

To create a new secret click the  button. Each secret consists of a unique name, optional description, secret type and authentication type. Depending on the selected secret and authentication type the additional input fields are different (for more details see the [Secret types](#) section). Once the secret is created it is visible in the secrets table that lists all secrets that you have access to, including the ones that have been shared with you.

Edit an existing secret

To edit an existing secret click the  icon on the row corresponding to the secret you want to edit, and then click *Edit*. This will open the edit menu bar where you can adapt the values of the secret. To store the changes click the *Save changes* button. Please note that the secret and authentication type cannot be changed. To change these you need to create a new secret.

Delete a secret

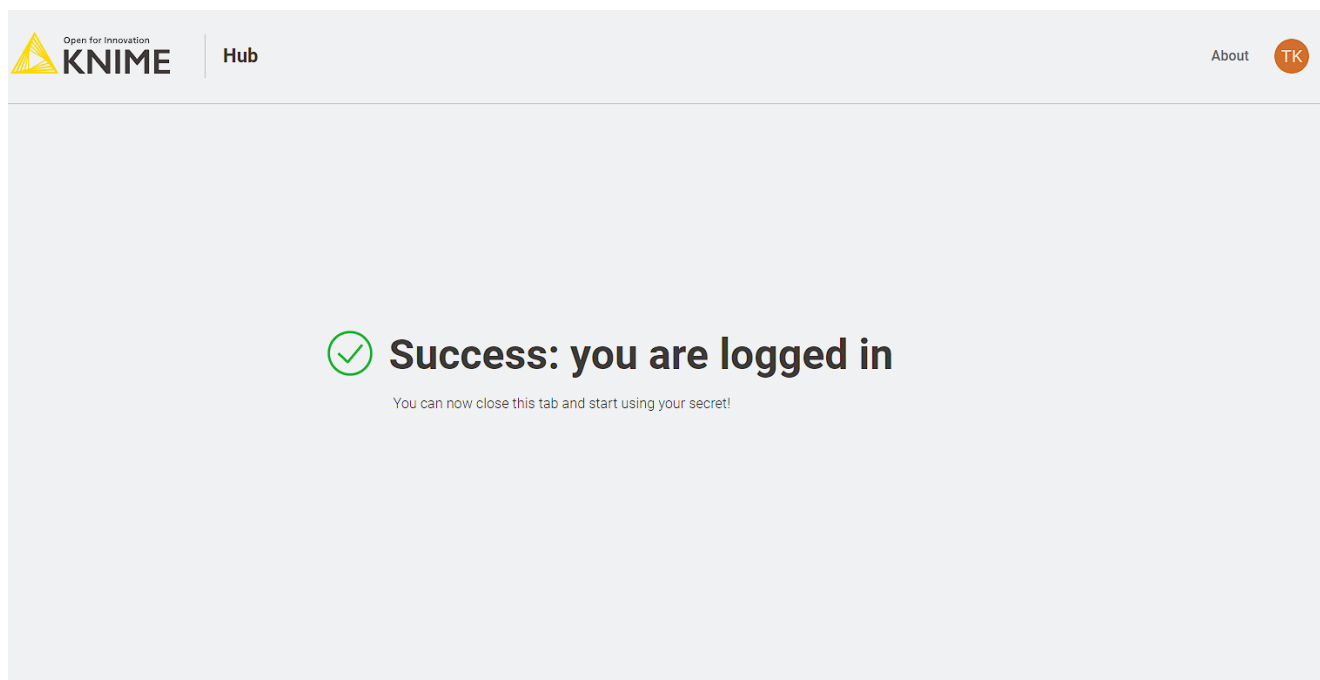
To delete an existing secret click the  icon on the row corresponding to the secret you want to edit, and then click *Delete*. To prevent accidental deletes you are prompted to enter the name of the secret. Once you have entered the secret name you can click the *I understand the consequences, delete secret permanently* button to delete the secret.

Interactive login

Some secret types, e.g. **interactive generic OAuth2** and other secrets with interactive authentication type, require you to log into your account to acquire a session and refresh token. If you are not logged in, these secrets are marked in the secrets table as *Not consumable* in the status column.

To log in click the  icon on the row corresponding to the secret you want to log in, and then click *Login*. This will open up a new browser window/tab that shows the login page of the corresponding identity provider, e.g. Google, Microsoft, etc.

Once you are logged in you are redirected to the success page which you can close.



Manage access to team secrets

For team secrets, you can select *Manage access* from the menu that opens when clicking the  icon of the secret to share. This opens a side panel where you can type the name of the user you want to share the secret with.

When managing the access you can assign either the *Use* or *Edit* right. The use right allows only the use of the secret in a KNIME workflow via the Secrets Retriever node (for more details see the [Using secrets](#) section). The edit right instead, also allows the user to not only use the secret but also to change its properties or to delete it.



User secrets cannot be shared with other users or teams for security reasons.

Using secrets

Secrets can be used in KNIME workflows via the **Secrets Retriever** node. The node is part of the **KNIME Hub Additional Connectivity** extension and needs to be installed separately. To retrieve the available secrets the Secrets Retriever node requires a connection to KNIME Community Hub.

This can be done in two ways:

1. If the workflow is located in a **space**, you can simply double click the workflow to open it. This will open the workflow on its location on the KNIME Community Hub for you to use from your local installation of the KNIME Analytics Platform client. By doing so the node will use the existing connection to the Community Hub to retrieve the secrets.
2. If the workflow is not located on the Hub e.g. it is stored in a **local workspace**, you need to use the **KNIME Hub Authenticator** and connect it to the Secrets Retriever node via its **dynamic input port**.

Once the node has access to KNIME Community Hub you can open its dialog. In the dialog you can select any number of secrets that you have access to. Depending on the types of the selected secret the node will have different output ports e.g. a **flow variable** output port if you select a secret of type **credentials**. For more details about the supported secret types see the **Secret types** section.



For security reasons the retrieved secrets are not stored when the workflow is saved. Therefore, the node needs to be re-executed every time a workflow is opened.

During execution of a workflow the Secrets Retriever node retrieves the secret from the Hub using the rights of the user that executes the workflow. If that user has no right to use the secret the node will fail with a `Secret does not exist` error message.



Secrets are referenced from the Secrets Retriever node using their internal identifier and not their name. So changing the name of a secret in the KNIME Hub will not break the connection to it from the Secrets Retriever node.

Secret types

Secrets can have various types such as credentials, access tokens, OAuth2 session tokens or private key files. Each secret type can have various authentication types e.g. the credentials type can be a Username/Password or Password only authentication type.

Choosing the right secret type

Table 1. Overview of authentication methods for workflow execution and Deployment

Interactive and ad hoc workflow execution	Workflow deployments
• Interactive OAuth	• Service accounts • Security token • Key file
For legacy systems you can use the credentials secret type, but it is not recommended since any user with permission to use it or Hub administrators can view the secret in plain text.	

Interactive OAuth

When working interactively with a workflow in KNIME Analytics Platform or executing it ad hoc on the KNIME Hub, use **interactive OAuth** secrets.

Secret store currently supports interactive OAuth for the following providers:

- [Box](#)
- [Databricks](#)
- [Google](#)
- [Microsoft](#)
- Other providers via the [generic OAuth secret type](#).

Enhanced security: Interactive OAuth secrets offer enhanced security through multi-factor authentication (MFA) and scoped access to external systems. Additionally, when using interactive OAuth, secret store only exposes short-lived session tokens, which are generated using an internally stored refresh token, further reducing security risks.

Limitations: Refresh tokens may expire, and depending on your identity provider, you may

need to **log in** again to update them. For this reason, it is not recommended to use interactive OAuth secrets for deployed workflows such as data apps or scheduled executions. For these use cases, use **service accounts** instead.

Service accounts

Service accounts are a special type of account that belong to an application or a virtual machine (VM), not to an individual user.

Secret store supports service accounts for the following providers:

- **Box**
- **Databricks**
- **Google**
- **Microsoft**
- **Generic OAuth providers.**

If a service account is not available, consider using a **security token** or **key file**. Be aware, however, that these usually grant full access to the external system, so use them with caution.



For legacy systems you can use the **credentials secret type**. Keep in mind that while KNIME tries to hide credentials, any user with permission to use a credentials secret or Hub administrator can view it in plain text, making it less secure than the other options.

Box

The screenshot shows the KNIME Hub interface. On the left, there's a sidebar with the 'Documentation team' (DT) logo and a list of items: Spaces, Deployments, Execution resources, and Secrets. The main area displays 'Secrets of Documentation team' with a table of secrets. The table has columns: Name, Owner, Type, Status, Updated on, and Description. It shows secrets for 'Dev Team' and 'Documentation team'. A 'Create secret' dialog is open on the right, showing fields for Name, Description, Secret type (Box), Authentication type (Server authentication (client credentials)), and Client/App configuration (ID, Secret, Enterprise ID). The dialog has 'Cancel' and 'Create' buttons.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared...
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-on...
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f...

The Box secret type allows you to connect to **Box** e.g. in order to manage your files using the **Box File Handling Extension**. Prior creating one of these secret types you need to create a **Custom App** in Box. Depending on the selected **authentication type** of your App you must select one of the authentication types described in the following sections.

For each of the following Box secret types the **Secrets Retriever** node will return a Credential output port with the Box access token (for more details see the **Using secrets** section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for the **Box Connector** node which allows you to **manage your files** in Box.

User authentication

This type is used for the **user authentication type** in Box which supports **OAuth 2.0** based user authentication.



This authentication type is **only available** for **personal secrets** of KNIME Hub users and not for **admin secrets** and **team secrets**.

This type requires you to log in to Box to obtain a valid access token prior using the secret. For more details on how to log in see the **Interactive login** section.

When you **setup** the Box App you have to add the following **redirect URI** to the **OAuth 2.0 Redirect URI** section in the **Apps Configuration** page <https://api.hub.knime.com/oauth2-flows/callback>. Note that the hostname must be prefixed with **api.** in the redirect URI. For more details on how to setup a User authentication (OAuth 2.0) App see the **Box**

[documentation](#).

For this secret type you can specify:

- Client/App ID: is the client ID of the application specified in the *OAuth 2.0 Credentials* section of your *App Configuration*
- Client/App Secret: is the client secret of the application specified in the *OAuth 2.0 Credentials* section of your *App Configuration*

Server authentication

This type is used for the [server authentication \(client credentials grant\)](#) which is recommended to use for deployed KNIME workflows. For more details on how to setup an application with Client Credentials Grant in Box click [here](#).

For this secret type you can specify:

- Client/App ID: is the client ID of the application specified in the *OAuth 2.0 Credentials* section of your *App Configuration*
- Client/App Secret: is the client secret of the application specified in the *OAuth 2.0 Credentials* section of your *App Configuration*
- Enterprise ID: is the *Enterprise ID* as displayed in the *General Settings* page of your App

Credentials

The screenshot shows the KNIME Community Hub interface. On the left, there's a sidebar with the 'Documentation team' (DT) profile and a list of items: Spaces, Deployments, Execution resources, and Secrets. The main area displays 'Secrets of Documentation team' with a table of secrets. On the right, the 'Create secret' dialog is open, showing fields for Name, Description, Secret type, Authentication type, and Credentials elements.

Secrets of Documentation team

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is a secret for the Corporate DB
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared HR DB
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-only HR DB
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login for HR Development

Create secret

Here you will be able to create your secret. Secrets allow you to store your logins in a safe way.

Secret basics

Name: Finance DB

Description: Login for finance DB

Secret type

Type of secret: Credentials

Authentication type: Username/Password

Credentials elements

Username: finance_user1

Password: [masked]

Buttons: Cancel, Create

Credentials are the most basic types of secrets. They are divided into the two authentication types: *Username/Password* and *Password only*. Whereby the *Username/Password* type stores a username and a password such as a database login and the *Password only* type stores a password only such as an API key or access token.

Independent of the number of selected credentials secret types the **Secrets Retriever** node will return a single flow variable output port with a credentials flow variable for each selected credentials secret (for more details see the **Using secrets** section). To better distinguish the different credentials variables, you can specify the name for each variable in the node dialog.

Credentials variables are supported by a wide range of KNIME nodes where they can be assigned in the node dialog to a username and password or token input field via the corresponding **flow variable** button.

Databricks

The screenshot shows the KNIME Hub interface. On the left, there's a sidebar with the 'Documentation team' (DT) logo and a list of items: Spaces, Deployments, Execution resources, and Secrets. The main area displays 'Secrets of Documentation team' with a table of secrets. On the right, a 'Create secret' dialog is open, showing fields for Name, Description, Secret type, Authentication type, Databricks workspace URL, and Token.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared...
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-on...
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f...

Create secret

Here you will be able to create your secret. Secrets allow you to store your logins in a safe way.

Secret basics

Name: Databricks

Description: Databricks account

Secret type

Type of secret: Databricks

Authentication type: Personal Access Token

Databricks workspace URL

Workspace URL:

Token

Personal access token:

Buttons: Cancel, Create

The Databricks secret type allows you to connect to various Databricks services e.g. manage your files in [Unity Catalog volumes](#) or orchestrate your [Spark jobs](#) in the Databricks Runtime. In order to use these services you need to install the [KNIME Databricks Integration extension](#). As a starting point for working with Databricks from within KNIME see the [KNIME for Databricks Users collection](#) or the [KNIME Databricks Integration User Guide](#).

For each of the following Databricks secret types the [Secrets Retriever](#) node will return a Credential output port with the [Databricks access token](#) (for more details see the [Using secrets](#) section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for the [Databricks Workspace Connector](#) which is the starting point to connect to different Databricks services such as [managing your files in Unity Catalog volumes](#) using the [Databricks Unity File System Connector](#).

The following sections describe the different supported authentication types. For more details on which authentication type to choose see the [Databricks documentation](#).

Interactive (OAuth2 U2M)

This type is used for interactive authentication using your personal Databricks login. This authentication type is also called [OAuth U2M](#) or user-to-machine flow in the [Databricks documentation](#).



This authentication type is **only available** for **personal secrets** of KNIME Hub users and not for **admin secrets** and **team secrets**.

Prior creating an interactive secret you need to create an [App connection](#) in your Databricks [account console](#). When you setup the connection enter the following [redirect URI](#) into the *Redirect URLs* section <https://api.hub.knime.com/oauth2-flows/callback>. In the *Access scopes* section select *All APIs*. If you select the *Generate a client secret* option, you need to select *Confidential* as type and enter the client ID and client secret into the corresponding fields of the Client configuration section of your new secret type. Otherwise select *Public* as type and enter only the client ID. For more details about how to setup App connections via the Databricks CLI or UI see the [Databricks documentation](#).

This type requires you to login to Databricks to obtain a valid [access token](#) prior using the secret. For more details on how to log in see the [Interactive login](#) section.

For this secret type you can specify:

- Workspace URL: the URL of the [Databricks workspace](#) you want to work with, for example <https://dbc-a1b2345c-d6e7.cloud.databricks.com>
- Client configuration type: Select *Confidential* if you have selected the *Generate a client secret* option when creating the [App connection](#) in Databricks. Otherwise select *Public*.
- Client ID: is the Client ID of the created [App connection](#)
- Client secret: only available if you choose the *Confidential* client type and only necessary if you selected the *Generate a client secret* option when creating the [App connection](#) in Databricks

Service Principal (OAuth2 M2M)

This type is used for a Databricks service principal which is recommended to use for deployed KNIME workflows. This authentication type is also called [OAuth M2M](#) or machine-to-machine flow in the [Databricks documentation](#). For more details on how to create a Databricks service principal see the [Databricks documentation](#).

For this secret type you can specify:

- Workspace URL: the URL of the [Databricks workspace](#) you want to work with, for example <https://dbc-a1b2345c-d6e7.cloud.databricks.com>
- Client ID: is the Client ID that can be created as described in the [Databricks documentation](#)
- Client secret: is the Secret that can be created as described in the [Databricks documentation](#)

Personal Access Token

This type is used for workspace users or service principals using a **personal access token (PAT)**. For more details on how to create a personal access token see the [Databricks documentation](#).



Databricks recommends using OAuth instead of PATs for user account client authentication and authorization due to the improved security OAuth has. To use OAuth, select one of the other authentication types.

For this secret type you can specify:

- **Workspace URL:** the URL of the [Databricks workspace](#) you want to work with, for example <https://dbc-a1b2345c-d6e7.cloud.databricks.com>
- **Personal access token:** is the token generated as described in the [Databricks documentation](#)

File

The screenshot shows the KNIME Hub interface. On the left, there's a sidebar with the 'Documentation team' (DT) logo and a list of items: Spaces, Deployments, Execution resources, and Secrets. The main area displays 'Secrets of Documentation team' with a table of secrets. A modal titled 'Create secret' is open on the right, showing fields for Name, Description, Secret type (File), Authentication type (File), and File management (Select file, snowflake.pem). The 'Create' button is highlighted in yellow.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared...
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-o...
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f...

The file credential type supports storing of arbitrary files such as api key or certificate files with a maximum size of 10 kilobytes.

Independent of the number of selected file secret types the **Secrets Retriever** node will return a single flow variable output port with a **path flow variable** for each selected file secret (for more details see the [Using secrets](#) section). To better distinguish the different file variables,

you can specify the name for each variable in the node dialog.

Path variables are supported by a wide range of KNIME nodes where they can be assigned in the node dialog to a file path via the corresponding **flow variable** button.

Generic OAuth2

The screenshot shows the KNIME Hub interface. On the left, the 'Documentation team' (DT) profile is visible with a sidebar containing 'Spaces', 'Deployments', 'Execution resources', and 'Secrets'. The main area displays 'Secrets of Documentation team' with a table of secrets. A 'Create secret' dialog is open on the right, showing the 'Generic OAuth2' secret type and various configuration options.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared...
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-on...
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f...

Create secret

Here you will be able to create your secret. Secrets allow you to store your logins in a safe way.

Secret type

Type of secret: Generic OAuth2

Authentication type

Client credentials

Endpoints configuration

Token endpoint URL:

Client/App configuration

ID:

Secret:

Scopes of access

+ Add scope

Advanced settings

Cancel Create

The generic OAuth2 secret type allows you to connect to **OAuth2** compliant authentication providers for which we do not have a dedicated secret type, e.g. Facebook, GitHub, Instagram, LinkedIn, Slack, and others. Depending on your use case and **grant type** you must select one of the authentication types described in the following sections.

For each of the following generic OAuth2 secret types the **Secrets Retriever** node will return a Credential output port with the OAuth2 access token (for more details see the **Using secrets** section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for several nodes such as the **REST nodes**.

Interactive

This type supports the (interactive) **OAuth 2.0 Authorization Code** grant flow. The auth code flow is used to obtain an access token via an interactive login. For more details on how to log in see the **Interactive login** section.



This authentication type is **only available** for **personal secrets** of KNIME Hub users and not for **admin secrets** and **team secrets**.

When you setup the auth code flow with your identity provider you need to use the following redirect URI: `https://api.hub.knime.com/oauth2-flows/callback`. Note that the hostname must be prefixed with `api.` in the redirect URI.

For this secret type you can specify:

- Authorization endpoint URL: is the authorization endpoint URL of the OAuth service
- Token endpoint URL: is the token endpoint URL of the OAuth2 service
- Client/App Type: is the application flow type to use either public or confidential
- Client/App ID: is the client/application ID sometimes called API key
- Client/App Secret: is the client/application secret to use (only available for confidential type)
- Scopes of access: is the list of scopes to request for the access token
- Token endpoint request method: is the HTTP method to use when requesting the access token from the token endpoint
- Client/App authentication method: specifies how to transfer Client/App ID and secret to the service endpoints. HTTP Basic Auth is the most common mechanism, but some services expect these values to be part of the form-encoded request body.
- Use PKCE: if selected a **Proof Key Code Exchange** is performed which improves the security

Client credentials

This type supports the **client credentials** grant flow. The client credentials grant is used to obtain an access token on behalf of an application/client, without having the context of a user.

For this secret type you can specify:

- Token endpoint URL: is the token endpoint URL of the OAuth2 service
- Client/App ID: is the client/application ID sometimes called API key
- Client/App Secret: is the client/application secret to use
- Scopes of access: is the list of scopes to request for the access token
- Token endpoint request method: is the HTTP method to use when requesting the access token from the token endpoint

- Client/App authentication method: specifies how to transfer Client/App ID and secret to the service endpoints. HTTP Basic Auth is the most common mechanism, but some services expect these values to be part of the form-encoded request body.
- Additional request fields: are any additional request body fields that should be added to the token endpoint request

Username/Password

This type supports the **OAuth 2.0 resource owner password credentials (ROPC)** grant flow.



The ROPC grant is considered legacy and does not support 2FA/MFA. Usage of this grant is discouraged and the client credentials grant should be used instead.

For this secret type you can specify:

- Token endpoint URL: is the token endpoint URL of the OAuth2 service
- Username: is the username to use
- Password: is the secret to use
- Client/App Type: is the application flow type to use either public or confidential
- Client/App ID: is the client/application ID sometimes called API key
- Client/App Secret: is the client/application secret to use (only available for confidential type)
- Scopes of access: is the list of scopes to request for the access token
- Token endpoint request method: is the HTTP method to use when requesting the access token from the token endpoint
- Client/App authentication method: specifies how to transfer Client/App ID and secret to the service endpoints. HTTP Basic Auth is the most common mechanism, but some services expect these values to be part of the form-encoded request body.

Google

The screenshot shows the KNIME Hub interface. On the left, there's a sidebar with the 'Documentation team' (DT) logo and a list of items: Spaces, Deployments, Execution resources, and Secrets. The main area displays 'Secrets of Documentation team' with a table of secrets. A 'Create secret' dialog is open on the right, showing the 'Secret basics' section with 'BigQuery main account' as the secret name. The 'Secret type' is set to 'Google'. The 'Authentication type' is 'Service account'. The 'Authentication Key' section shows 'JSON' as the type and 'P12' as the format. The 'JSON file' section has a 'Select file' button. The 'Scopes of access' section shows 'Standard' as the scope type and 'Google BigQuery' as the selected scope. The 'Create' button is highlighted in yellow.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared...
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-on...
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f...

The Google secret type allows you to connect to various Google services e.g. manage your files in [Google Drive](#) via the [Google Connectors Extension](#) or [Google Cloud Storage](#) using the [Google Cloud Storage Extension](#) as well as working with your data in [Google BigQuery](#) using the [BigQuery Extension](#).

For each of the following Google secret types the [Secrets Retriever](#) node will return a Credential output port with the [Google access token](#) (for more details see the [Using secrets](#) section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for various nodes such as the [Google Driver Connector](#) which allows you to [manage your files](#) in [Google Drive](#) or the [Google BigQuery Connector](#) which allows to [manage your data](#) in [Google BigQuery](#).

Interactive

This type is used for interactive authentication using your personal Google login.



This authentication type is **only available** for **personal secrets** of KNIME Hub users and not for **admin secrets** and **team secrets**.

Prior creating an interactive secret you need to create an [OAuth consent screen](#) and [OAuth client ID](#) of type Web application. When you setup the Web application you don't need to specify anything in the *Authorized JavaScript origins* section. But you have to add the following [redirect URI](#) to the *Authorized redirect URIs* section in the OAuth client ID configuration page <https://api.hub.knime.com/oauth2-flows/callback>. For more details

about authentication in general and further details how to setup the OAuth consent screen and client id see the [Google documentation](#).

This type requires you to login to Google to obtain a valid [access token](#) prior using the secret. For more details on how to log in see the [Interactive login](#) section.

For this secret type you can specify:

- Client/App ID file: is the [OAuth client ID](#) secret file (for an example file click [here](#))
- Scopes of access: are the scopes to request during login (for more details see the [OAuth2 scopes](#) section)

Service account

This type is used for a [service account](#) which is recommended to use for deployed KNIME workflows. For more details on how to setup a service account in Google click [here](#).

For this secret type you can specify:

- Authentication key type: is either JSON (recommended) or P12 format
- JSON or P12 file: depending on the selected key type please either upload your JSON or P12 [key file](#) (See the [Google documentation](#) on how to create one)
- Service account email: is the email address of the service account (only available for P12 type)
- Scopes of access: are the scopes to request during login (for more details see the [OAuth2 scopes](#) section)

Standard OAuth2 scopes for Google services

This section lists the different scopes for most common Google services you can access from within KNIME Analytics Platform.

For more details and a complete list of all the available scopes see the [Google documentation](#). To use one of the common services mentioned below copy the URL next to the service and paste it into the *Scopes of access* section of the secret.

- Google Analytics (read-only): <https://www.googleapis.com/auth/analytics.readonly>
- Google BigQuery: <https://www.googleapis.com/auth/bigquery>
- Google Drive (read-only): <https://www.googleapis.com/auth/drive.readonly>

- Google Drive: <https://www.googleapis.com/auth/drive>
- Google Sheets (read-only): <https://www.googleapis.com/auth/spreadsheets.readonly> and <https://www.googleapis.com/auth/drive.readonly>
- Google Sheets: <https://www.googleapis.com/auth/spreadsheets> and <https://www.googleapis.com/auth/drive.readonly>

Microsoft

The screenshot shows the KNIME Hub interface. On the left, the 'Documentation team' sidebar is visible with a 'Secrets' tab selected. The main area displays 'Secrets of Documentation team' with a table of secrets. A 'Create secret' modal is open on the right, showing the configuration for a Microsoft secret.

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is a secret for the Corporate DB.
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared HR DB
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-only HR DB
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login for HR Development

Create secret

Here you will be able to create your secret. Secrets allow you to store your logins in a safe way.

Secret type

Type of secret: Microsoft

Authentication type

Application/Service principal

Domain configuration

Tenant ID/Domain:

Client/App configuration

ID:

Client application secret:

Scopes of access

Scope type: Standard Custom

Sharepoint

Cancel Create

The Microsoft secret types allow you to connect to the Microsoft/Azure cloud platform with nodes from the following extensions:

- **KNIME Office 365 Connectors**
- **KNIME SharePoint List**
- **KNIME Azure Cloud Connectors**
- **KNIME Power BI Integration**
- **KNIME Snowflake Integration**
- **KNIME Database (Microsoft SQL Server Connector node)**
- **KNIME REST Client Extension**

For each of the following secret types the Secrets Retriever node will return a Credential output port that contains either an OAuth2 access token, or an Azure Storage credential (for

more details see the [Using secrets](#) section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for nodes from the above extensions.

Interactive

This type supports an interactive login into the Microsoft/Azure cloud with your Microsoft identity.



This authentication type is **only available** for [personal secrets](#) of KNIME Hub users and not for [admin secrets](#) and [team secrets](#).

Prior to using the secret, you need to perform an [Interactive login](#) to obtain a valid access token. Note that at some point the login will expire and a new login is necessary, hence this secret type is not well-suited for scheduled or otherwise automated workflows on KNIME Hub.

As a prerequisite to configuring the secret, an App in Azure Entra ID ([formerly Azure Active Directory](#)) needs to be registered (see [How-tos](#) section).

For this secret type you can specify:

- Client/App configuration:
 1. Select:
 - a. *Public* if you registered a *Public* App in Azure Entra ID. This means that you selected *Public client/native (mobile & desktop)* under *Redirect URI* when [creating Azure App](#).
 - b. *Confidential* if you registered a *Confidential* App in Azure Entra ID. This means that you selected *Web* under *Redirect URI* when [creating Azure App](#).
 2. Enter the Application ID of the previously registered Azure App
- Scopes of access: enter a list of scopes, which limit what the resulting secret can be used for, e.g. only to access SharePoint; during the interactive login you may have to consent to the requested scopes (for more details see the [OAuth2 scopes](#) section).
- Authorization endpoint: either use the default URL, or enter a custom one, which allows to sign into a specific Azure tenant



In technical terms, the login is based on the [OAuth 2.0 Authorization Code](#) flow. The interactive login allows KNIME Hub to obtain and store temporary access and refresh tokens on behalf of the user. KNIME Hub refreshes and returns the acquired access token whenever the secret is used in a workflow. The selected scopes correspond to [delegated permissions](#) in Microsoft/Azure. Consult the respective [how-to](#) for more information on how to correctly set up an Azure App.

Application/Service principal

This type supports authenticating as an application or service principal in the Microsoft/Azure cloud. This is well-suited for scheduled or otherwise automated workflows on KNIME Hub, where no user is present to interactively login.

As a prerequisite, an App in Azure Entra ID ([formerly Azure Active Directory](#)) needs to be registered. Please see the respective [how-to](#).

For this secret type you can specify:

- Domain configuration: specify the Azure tenant to access, either in ID format, e.g. `faa16e7e-a95d-4117-b2c7-06ffc6e68acb`, or as a domain name, e.g. `contoso.onmicrosoft.com`
- Client ID and secret: enter the client ID and secret of the previously registered Azure App
- Scopes of access: enter a list of scopes, which limit what the resulting secret can be used for, e.g. only to access SharePoint (for more details see the [OAuth2 scopes](#) section)



In technical terms, the authentication is based on the [OAuth 2.0 Client Credentials](#) flow. KNIME Hub requests a new access token whenever the secret is used in a workflow. The selected scopes correspond to [application permissions](#) in Microsoft/Azure. Consult the respective [how-to](#) for more information on how to correctly set up an Azure App and application permissions.

Username/Password

This type supports authenticating as a user in the Microsoft/Azure cloud. It can be used for scheduled or otherwise automated workflows on KNIME Hub, where no user is present to interactively login. However, note that this authentication type is [discouraged by Microsoft](#). It

does not support accounts 2FA/MFA and has further limitations.

As a prerequisite, an App in Azure Entra ID (**formerly Azure Active Directory**) needs to be registered. Please see the respective **how-to**.

For this secret type you can specify:

- Credentials: specify your Microsoft/azure username and password.
- Client/App configuration: enter the client ID of the previously registered Azure App
- Scopes of access: enter a list of scopes to limit what the resulting secret can be used for, e.g. only to access SharePoint (for more details see the **OAuth2 scopes** section)
- Authorization endpoint: either use the default URL, or enter a custom one, which allows to sign into a specific Azure tenant



In technical terms, the authentication is based on the **OAuth 2.0 Resource Owner Password Credentials flow**. KNIME Hub requests a new access token whenever the secret is used in a workflow. The selected scopes correspond to **delegated permissions** in Microsoft/Azure. Consult the respective **how-to** for more information on how to correctly set up an Azure App and delegated permissions.

Azure Storage shared access signature (SAS)

This type allows to authenticate against Azure Blob Storage/Data Lake Storage Gen2 with a *Shared Access Signature (SAS)*. A SAS grants restricted and time-limited access to an Azure Storage container or objects within. See **here** for further documentation.

For this secret type you only need to specify a SAS URL. Consult the respective **how-to** for how to create a SAS URL.

Azure Storage shared key

This type allows to authenticate against Azure Blob Storage/Data Lake Storage Gen2 with a *shared key*. The shared key grants unrestricted access to an Azure Storage account and all containers within. See **here** for further documentation.



Microsoft/Azure recommends to not use shared key authentication, as it provides unrestricted access to an Azure Storage account and all containers within. Any of the other authentication types in this section can be used instead.

For this secret type you need to specify:

- Storage account: enter the unique name of the storage account
- Shared key: the shared key, which can be located as described [here](#)

Standard OAuth2 scopes for Azure services

This section lists the different scopes for most common Azure services you can access from within KNIME Analytics Platform. For more details and a complete list of all the available scopes see the [Microsoft documentation](#). To use one of the common services mentioned below copy the scope next to the service and paste it into the *Scopes of access* section of the secret.

- Sharepoint files and list items (Read): `Sites.Read.All` This permission allows the token to be used for reading files and list items stored on SharePoint Online. Note that access to any specific SharePoint site needs to be additionally granted to the user by that site.
- Sharepoint files and list items (Read/Write): `Sites.ReadWrite.All` This permission allows the token to read and write files as well as list items stored on SharePoint Online. This includes creating and deleting files, but not lists. Note that access to any specific SharePoint site needs to be additionally granted to the user by that site.
- Sharepoint files, lists and list items (Read/Write): `Sites.Manage.All` This permission allows the token to read and write files, lists as well as list items stored on SharePoint Online. This includes creating and deleting files as well as lists. Note that access to any specific SharePoint site needs to be additionally granted to the user by that site.
- User Groups (Read) (Requires admin consent): `Directory.Read.All` This permission is required to browse the Office 365 groups that the logged in user is a member of, when selecting a SharePoint team site to connect to. Note that this permission can only be granted by a Entra ID admin.
- User Groups (Read) (Limited): `User.Read` This permission is required to browse the Office 365 groups that the logged in user is a member of, when selecting a SharePoint team site to connect to. This permission does not require consent by an admin, but cannot access the human-readable names of Office 365 groups, hence only technical IDs will be displayed.
- Azure Blob Storage/Azure Data Lake Storage Gen2: Requests the [user_impersonation](#) permission for a specific Azure storage account. This permission allows the token to access data stored in that storage account. Note that access to any specific data in that account needs to be additionally granted to the user, before any access is possible.
- Azure SQL Database: Requests the [user_impersonation](#) permission. This permission

allows the token to access the Azure SQL API. Note that access to any specific databases/resources needs to be additionally granted to the user, before any actual access is possible.

- Power BI: `Dataset.ReadWrite.All` and `Workspace.Read.All`.

Salesforce

The screenshot shows the KNIME Hub interface. On the left, the 'Documentation team' (DT) profile is visible with a sidebar containing 'Spaces', 'Deployments', 'Execution resources', and 'Secrets'. The main area displays 'Secrets of Documentation team' with a table of secrets. A 'Create secret' modal is open on the right, showing the 'Secret type' as 'Salesforce' and the 'Authentication type' as 'Username/Password'. The modal also includes fields for 'Connected App' ID, 'Secret', and 'User credentials' (Username, Password, Security Token).

Name	Owner	Type	Status	Updated on	Description
Dev Team					
Corporate DB	Dev Team	Credentials	✓	Nov 8, 2024 2:25 PM	This is...
Documentation team					
Shared HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:25 PM	Shared
HR DB	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Read-on
HR Development	Documentation team	Credentials	✓	Jan 8, 2025 4:24 PM	Login f

The Salesforce secret type allows you to connect to **Salesforce** by using the nodes from the **Salesforce Extension**.

For each of the following Salesforce secret types the **Secrets Retriever** node will return a Salesforce Credential output port with the Salesforce access token (for more details see the **Using secrets** section). Each selected secret will result in a dedicated Credential output port. This port can be used as input for the **Salesforce Connector** node which is the starting node to access your data in Salesforce.

Prior creating one of these secret types you need to create a **Connected App** in Salesforce with **enabled OAuth Settings**. In the **API (Enable OAuth Settings)** section you need to provide the following Callback URL `https://api.hub.knime.com/oauth2-flows/callback`. Note that the hostname must be prefixed with `api.` in the redirect URI. For more information about the used OAuth Authorization Flows in general see the **Salesforce documentation**.

Interactive

This type is used for the [OAuth 2.0 Web Server Flow](#) in Salesforce which implements the [OAuth 2.0 authorization code grant type](#).



This authentication type is **only available** for **personal secrets** of KNIME Hub users and not for **admin secrets** and **team secrets**.

This type requires you to log in to Salesforce to obtain a valid access token prior using the secret. For more details on how to log in see the [Interactive login](#) section.

For this secret type you can specify:

- Connected App ID: is the consumer key of the connected application which can be viewed as described [here](#)
- Connected App Secret: is the consumer secret of the connected application which can be viewed as described [here](#)
- Salesforce Instance: is the Salesforce instance type either *Production* or *Sandbox* for testing

Username/Password

This type is used for the [OAuth 2.0 Username-Password Flow](#) which is recommended to use for deployed KNIME workflows.

For this secret type you can specify:

- Connected App ID: is the consumer key of the connected application which can be viewed as described [here](#)
- Connected App Secret: is the consumer secret of the connected application which can be viewed as described [here](#)
- Username: is the username
- Password: is the password
- Security Token: is the optional security token
- Salesforce Instance: is the Salesforce instance type either *Production* or *Sandbox* for testing

How-tos

Create Azure App for user authentication

This how-to describes how to set up an Azure App, so that it can be used with secrets of type **Microsoft – Interactive** and **Microsoft – Username/Password**.

Prerequisites:

- Permission to register apps in Azure Entra ID. Azure admins generally have this permission, but it can also be provided through roles such as the *Application Administrator* or *Application Developer* (see [here](#)).

Steps:

1. Log into the **Azure Portal**
2. Navigate to *Azure Entra ID* → *App registrations*
3. Register a new app:
 - a. Click *New registration*
 - b. Enter a name, and choose the supported account type (*Single tenant* in most cases)

Microsoft Azure Search resources, services, and docs (G+)

Home > App registrations >

Register an application

Name

The user-facing display name for this application (this can be changed later).

KNIME App

Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (Single tenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client/native (mobile ... [▼](#)

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

c. Under *Redirect URI*:

i. Select the platform called either:

A. *Web*, or

B. *Public client/native (mobile & desktop)*

ii. Enter `https://api.hub.knime.com/oauth2-flows/callback`. Note that the hostname must be prefixed with `api.` in the redirect URI.

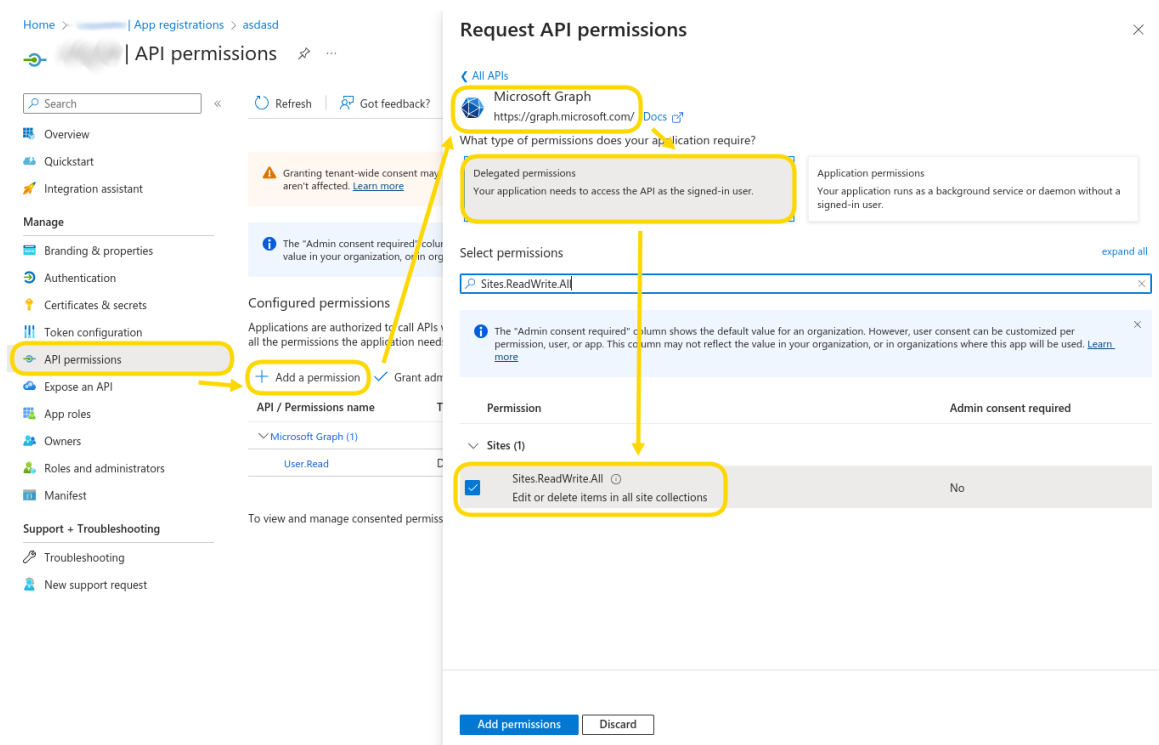
d. Click *Register*

4. Add delegated permissions:

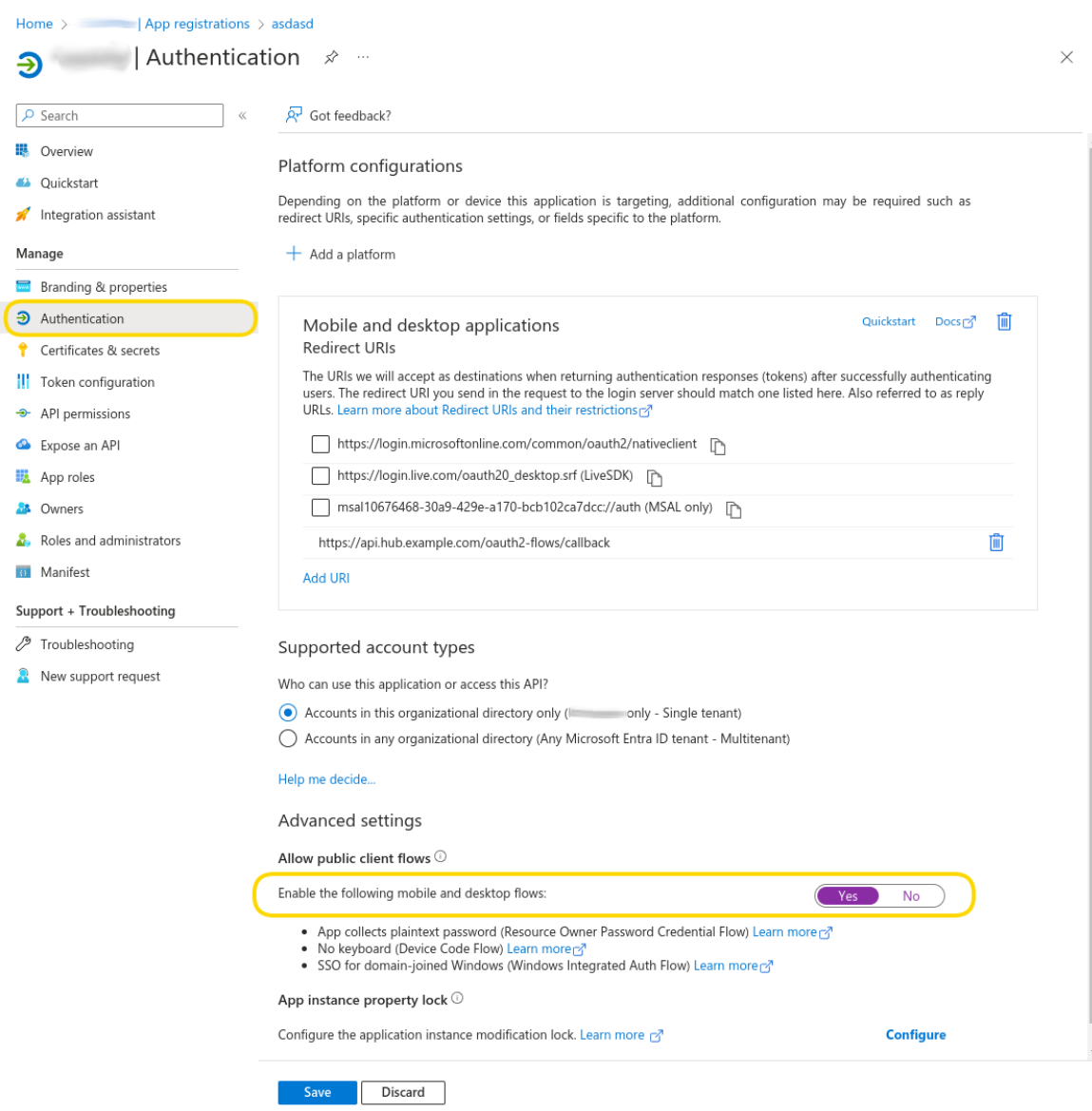
a. In your app, navigate to *API permissions*

b. Click *Add a permission* to add the necessary permissions that the app should have. For example, to allow read/write access to SharePoint choose *Microsoft Graph* → *Delegated permissions* → *Sites* → *Sites.ReadWrite.All*. The required permissions depend on how you plan to use the app in KNIME workflows (for a list of common scopes see the [OAuth2 scopes](#) section).

c. Click *Add permissions*



- d. If shown, click *Grant admin consent for ...* to give admin consent for all added API permissions. In this case users will not be prompted for consent during interactive authentication anymore.
5. Only if you need to use the app with secrets of type **Microsoft — Username/Password** (discouraged):
 - a. In your app, navigate to *Authentication* → *Allow public client flows*
 - b. Set the slider to Yes
 - c. Click Save



- d. Navigate to *API permissions*, add all required API permissions and consent to them as in step 6 above.
6. Retrieve the application ID:
 - a. In your app, navigate to *Overview*
 - b. Copy the *Application (client) ID*
7. In your KNIME Hub secret(s):
 - a. Paste the application ID into the *Client/App ID* field

i

API permissions, also called scopes, act as a upper limit on what can be done with the acquired access token. On top of that, most services in the Microsoft/Azure platform impose **additional** permission checks based on the user's roles and groups. Examples are **SharePoint permissions** or **Azure RBAC**.

Create Azure App for application/service principals

This how-to describes how to set up an Azure App, so that it can be used with secrets of type **Microsoft — Application/Service principal**.

Prerequisites:

- Permission to register apps in Azure Entra ID. Azure admins generally have this permission, but it can also be provided through roles such as the *Application Administrator* or *Application Developer* (see [here](#)).

Steps:

1. Log into the [Azure Portal](#)
2. Navigate to *Azure Entra ID* → *App registrations*
3. Register a new app:
 - a. Click *New registration*
 - b. Enter a name, and choose the supported account type (*Single tenant* in most cases). A redirect URI is not required.
 - c. Click *Register*
4. Add application permissions:
 - a. In your app, navigate to *API permissions*
 - b. Click *Add a permission* to add the necessary permissions that the app should have. For example, to allow read/write access to SharePoint choose *Microsoft Graph* → *Application permissions* → *Sites* → *Sites.ReadWrite.All*. The required permissions depend on how you plan to use the app in KNIME workflows (for a list of common scopes see the [OAuth2 scopes](#) section).
 - c. Click *Add permissions*
 - d. Click *Grant admin consent for ...* to give admin consent for all added API permissions.
5. Create an application secret:
 - a. In your app, navigate to *Certificates & secrets* → *Client secrets*
 - b. Click *New client secret*
 - c. Enter a description and expiry
 - d. Click *Add*
 - e. Copy the *Value* of the newly created client secret, as it is only shown once after creation.

6. Retrieve the application ID:

- a. In your app, navigate to *Overview*
- b. Copy the *Application (client) ID*

7. In your KNIME Hub secret(s):

- a. Paste the application ID into the *Client/App ID* field
- b. Paste the client secret into the *Client/App Secret* field



API permissions, also called scopes, act as a upper limit on what can be done with the acquired access token. On top of that, some services in the Microsoft/Azure platform impose **additional** permission checks based on roles and group memberships of the service principal. An example of this is **Azure RBAC**.

Create Azure Storage SAS URL

This how-to describes how to create an Azure Storage SAS URL, so that it can be used with secrets of type **Azure Storage shared access signature (SAS)**. A SAS URL grants restricted and time-limited access to an Azure Storage container or objects inside a container.

Prerequisites:

- Permission to create SAS URLs, depending on the type of SAS to create. See [here](#) for an overview.

To create an Azure Storage SAS URL for a container

1. Log into the **Azure Portal**
2. Navigate to *Storage Accounts* → *Your Account* → *Containers* → *Your Container* → *Shared access tokens*
3. Enter signing method, permissions and expiry information
4. Click *Generate SAS token and URL*
5. Copy the *Blob SAS URL*
6. In your KNIME Hub secret(s): Paste the copied *Blob SAS URL* into the *SAS URL* field

The steps to create SAS URL for a specific object inside a container are similar. Navigate to the object, click on it and then choose *Generate SAS*.

Find the Azure Storage shared key

This how-to describes how to locate the shared key of an Azure Storage account, so that it can be used with secrets of type **Azure Storage shared key**. Such a key grants unrestricted access to an Azure Storage account and all containers within.

Prerequisites:

- Permission to view the account keys of the Storage account. Several roles as described [here](#) allow to view account keys.

The shared key for an Azure Storage container can be found as follows:

1. Log into the **Azure Portal**
2. Navigate to *Storage Accounts* → *Your Account* → *Access keys*
3. Copy one of the shown keys
4. In your KNIME Hub secret(s): Paste the copied key into the *Shared key* field

Architecture



Figure 1. KNIME Secrets Store Architecture on KNIME Community Hub. The Secret Store consists of three components: the Secret Store Service, the Key Management Service, which stores the key encryption key (KEK), and the Object Store, which stores encrypted secrets and their data encryption keys(DEK). All communication is secured via TLS.

The KNIME Secret Store consists of three main services:

- Secret Store Service
- Key Management Service
- Object Store

All communication is secured via **TLS encryption**.

- **Secret Store Service:** is the core component responsible for managing secrets in the KNIME Community Hub.
The Secret Store Service stores non-sensitive metadata related to each secret, such as the secret's name and description, in a relational database.
The Secret Store Service encrypts sensitive data using envelope encryption.
- **Key Management Service:** Envelope encryption is implemented using the [AWS Encryption SDK](#) and the Key Management Service.
- **Object Store:** Once the secret is encrypted it is stored in the Object Store with server-side encryption.

Envelope encryption is a [hybrid cryptographic system](#) used by major cloud providers like

Amazon, Microsoft, and Google.

It uses two types of keys:

1. A unique data encryption key (DEK) for each secret
2. A key encryption key (KEK), which is rotated periodically for enhanced security.

KNIME AG
Talacker 50
8001 Zurich, Switzerland
www.knime.com
info@knime.com