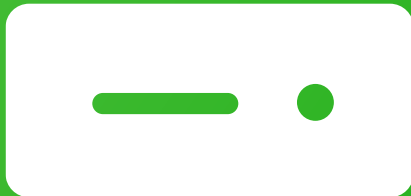


POWER AUTOMATE CONNECTOR

DOCUMENTATION

Version 2.1



automate your documents
integrate your data

dox42

dox42 GmbH | Vegagasse 5/2, A-1190 Vienna
IBAN: AT571420020010936536 | BIC: BAWAATWW | Handelsgericht Wien | FN: 393153 t | UID: ATU67845433

Content

1	Introduction	3
2	Using the dox42 Custom Connector Method 1 (import via Github)	4
2.1	Security	5
2.2	Definition and Test – Check Policy	7
2.3	Sharing is Caring	8
2.4	dox42 Custom Connector Example Flow	8
3	Using the Custom Connector Method 2 (create from scratch)	11
4	Using the dox42 Premium Connector, Method 1	17
4.1	Generating the ID Token	18
4.2	Setting up an Azure Key Vault	18
4.3	Configuring the Flow	19
5	Using the dox42 Premium Connector, Method 2	26
5.1	Generating the Access Token	27
5.2	Configuring the Flow	28
6	Running the Flow	30

1 Introduction

The dox42 Power Automate Premium Connector provides a connection to the very extensive dox42 REST API and allows you to automatically generate documents in a flow with data from all sources like D365, SharePoint or others. You can also use it from Azure Logic Apps and Power Apps. The connector is available as a Premium Connector or alternatively as a Custom Connector, which can be imported via GitHub or created yourself. The authentication is token based and using the Premium Connector the authentication happens outside the connector and we need to create Access/ID tokens within our Flows. With the Custom Connector the authentication happens within the connector, and you add all necessary Azure Active Directory information to the Security tab of your connector.

To use dox42 from Power Automate, you also need a dox42 Server/dox42 Online license. For authentication purposes, you will need to register your dox42 Server or dox42 Online license as an application in Azure Active Directory. You can find more information on your dox42 Online configuration and set-up [here](#). If you do not have your licenses yet or would like to get a free trial email us at info@dox42.com.

Power Automate

Search for helpful resources

Environments
dox42 (default)

dox42 (Preview)

The dox42 connector provides a connection to the very extensive dox42 REST API and allows you to automatically generate documents in a flow with data from all sources, like D365, SharePoint or others.

[See documentation](#)
PREMIUM

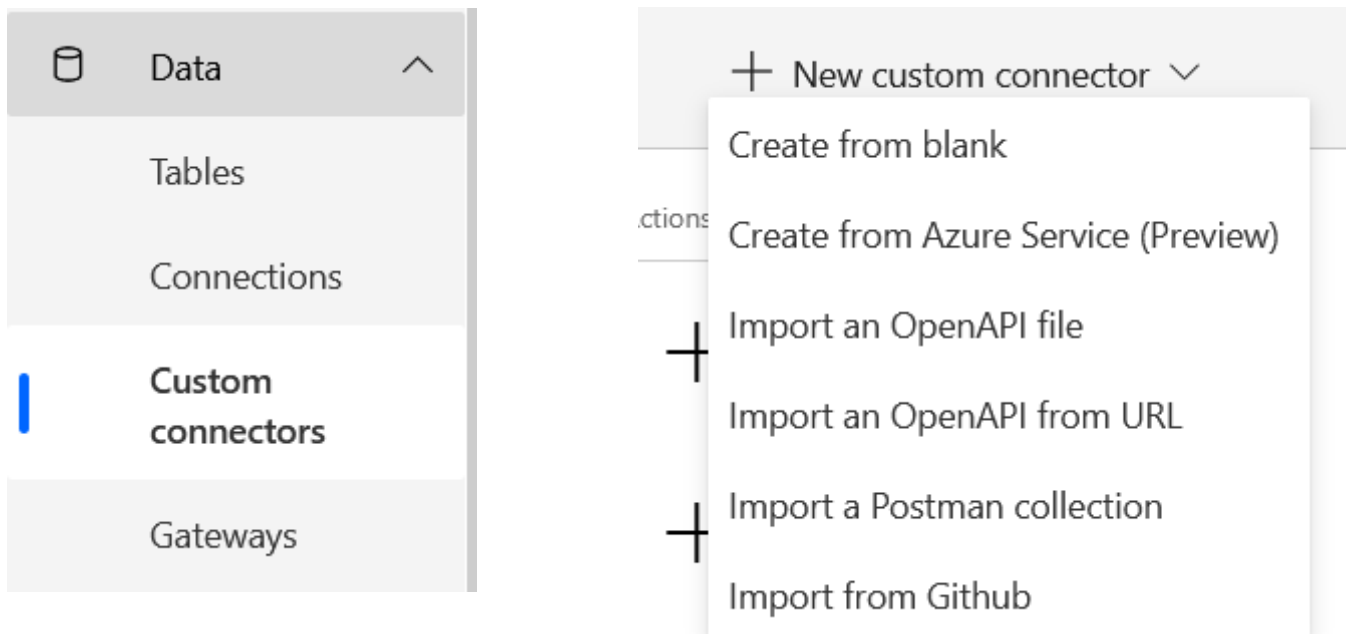
Triggers - A trigger is an event that starts a flow

There are currently no triggers for this connector. Help us decide which services and triggers to add. [UserVoice](#)

2 Using the dox42 Custom Connector Method 1 (import via Github)

Alternatively to using the dox42 Premium Connector, you can generate your documents from Microsoft Power Automate Flows via our custom connector. The main difference is that within the custom connector you add the authentication method within the Security tab of the connector, not in each Flow separately. **This is the most time efficient way of building your dox42 Power Automate Flows, hence we recommend this method.**


To trigger the dox42 Server from the dox42 custom connector, you need to import the dox42 custom connector from GitHub. It can be found in "certified connectors" in the "master" branch:

The image shows a screenshot of the 'Import from Github' dialog box. At the top, it says 'Import from Github' with a link 'Click here to explore the open source repository' and a GitHub logo. Below this, there are two radio buttons for 'Connector Type *': 'Certified' (selected) and 'Custom'. There are two dropdown menus: 'Branch' with 'master' selected, and 'Connector' with 'dox42' selected. At the bottom, there are two buttons: 'Continue' (blue) and 'Cancel' (grey).

Now you will see the dox42 connector in your list of custom connectors.

Next you need to add your dox42 Online tenant Name (or dox42 On-premise server) to this custom connector:

General information



Upload connector icon
Supported file formats are PNG and JPG. (< 1MB)

↑ Upload

Icon background color

A color to show behind the icon (e.g., '#007ee5')

Description

Trigger dox42 Online from MS Flow

Connect via on-premises data gateway [Learn more](#)

Scheme *

HTTPS HTTP

Host *

yourtenant.dox42.online

Base URL

/

Security →

2.1 Security

In the Security Tab select "OAuth 2.0" and "Azure Active Directory"

Security

Choose the authentication type and fill in the required fields to set the security for your custom connector. [Learn more](#)

Authentication type

Choose what authentication is implemented by your API *

OAuth 2.0

OAuth 2.0

Identity Provider

Azure Active Directory

Enable Service Principal support

Client ID *

915cc84b-12c4-4196-a7ea-xxxxx

Client secret *

.....

Authorization URL

https://login.microsoftonline.com

Tenant ID

common

Resource URL *

https://xxxxt.sharepoint.com

Enable on-behalf-of login

false

Scope

https://xxxxx.sharepoint.com/AllSites.Write

Redirect URL

https://global.consent.azure-apim.net/redirect/new-20blank-20connector-5f48af0b749eb... 📄

← General
Definition →

Enter your App ID of your dox42 (Online) Server App registration from Azure Active Directory ([see also dox42 AAD chapters in SharePoint and D365 CE documentation](#)), the Client Secret and the Resource URL of the SharePoint Online.

For the Scope please enter <https://YourTenant.SharePoint.com/AllSites.Write> or <https://YourTenant.SharePoint.com/AllSites.Read> for the tenant where you store your templates.

In the last field "Redirect URL" you receive a custom redirect URL for each dox42 Custom connector. Copy it and add it as a Web Redirect URL to your dox42 (Online) Server App registration in Azure Active Directory AAD Admin Center (next to all other Web Redirect URIs that you've configured for dox42 Online:

Search Got feedback?

- Overview
- Quickstart
- Integration assistant
- Diagnose and solve problems
- Manage
 - Branding & properties
 - Authentication**

Web

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. [Learn more about Redirect URIs and their restrictions](#)

This app has implicit grant settings enabled. If you are using any of these URIs in a SPA with MSAL.js 2.0, you should migrate URIs. →

https://global.consent.azure-apim.net/redirect/dox42-20-2d-201-5f48af0b749eb93eb9-5f719450877f2d9fe5-1	
https://global.consent.azure-apim.net/redirect/dox42-20-2d-201-5f48af0b749eb93eb9-5f719450877f2d9fe5	

2.2 Definition and Test – Check Policy

Since we have already configured the dox42 connector, this step is not necessary, though please double check if a policy is added to the connector.

In case there is none, please add the following:

Policy details

Name *

Define dox42 Service URL

Template * [Learn more](#)

Set host URL

Replaces host URL with the URL generated from the template.

Operations

List of actions and triggers to which the policy will apply to. If no operation is selected, this policy will apply to all operations.

dox42_Call

Url Template *

Specifies template from which host URL will be generated.

https://@headers('domainname')/dox42RestService.ashx?@queryParameters('querystring')

← Security Code (Preview) →

You can copy the URL template here:

https://@headers('domainname')/dox42RestService.ashx?@queryParameters('querystring')

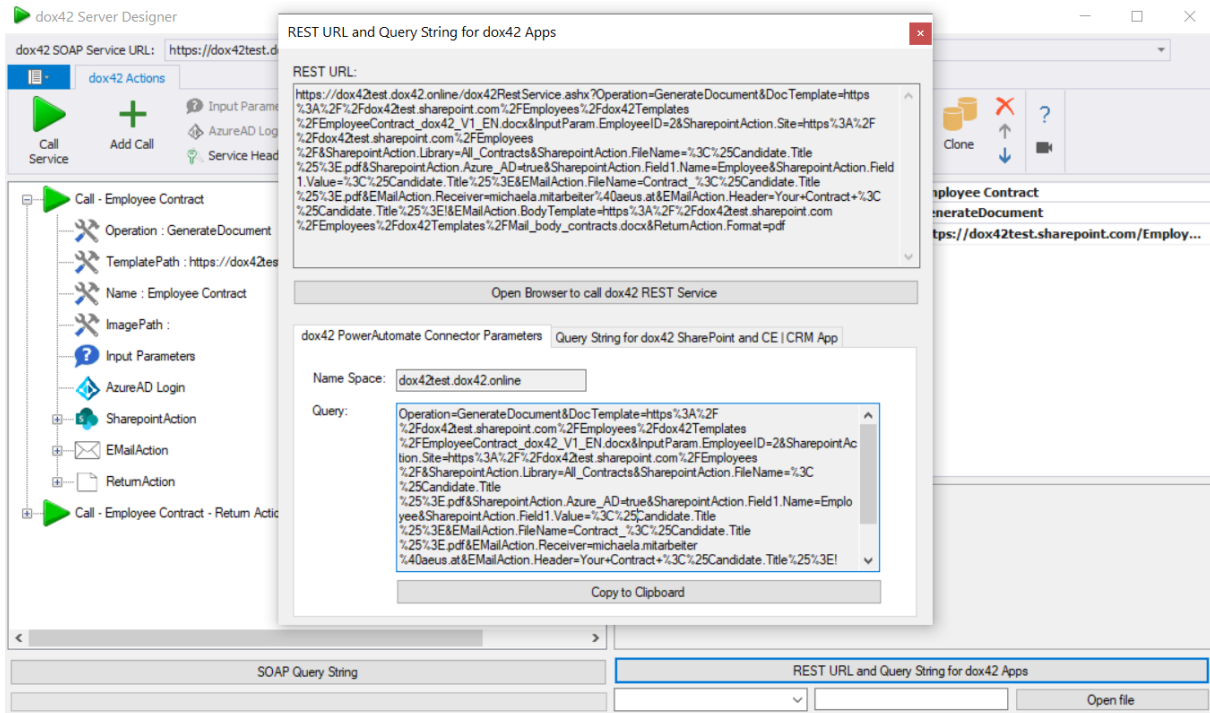
2.3 Sharing is Caring

Select the dox42 custom connector in the custom connectors list and share it with the people of your organization who should be able to use it. If you do not share, nobody but yourself will be able to use the custom connector.

2.4 dox42 Custom Connector Example Flow

The image shows a screenshot of a Power Automate flow configuration. The top section is a trigger titled "When an item is created or modified" with a SharePoint icon. It has two required fields: "Site Address" with the value "Employees - https://dox42test.sharepoint.com/Employees" and "List Name" with the value "Modern EmployeeContracts". Below these fields is a "Show advanced options" link. A downward arrow points to the second section, a "dox42 Service Call" action with a green play button icon. This action has three fields: "domainname" (empty), "querystring" (empty), and "accept" with the value "application/json".

Both "domainname" and "querystring" can be retrieved from the dox42 Server Designer (Version 1.0.1.4 and later) like this:



Configure a working config in the dox42 Server Designer and navigate to “REST URL and Query String for dox42 Apps” in the “dox42 Power Automate Connector Parameters” you will find the correct parameters for the dox42 connector. “Name Space” represents the “domainname” parameter and “Query” the “querystring” parameter.

Now you can run your dox42 server call in Power Automate using the dox42 custom connector.

2.4.1 dox42 Custom Connector Example Flow with POST Call

You can also use your dox42 custom connector to run POST calls. This can be useful if you send data via a Request Body. Choose the option of a POST Request in your Flow and add the Request Body portion of your call in the “Request Body” field, instead of the Query Portion field.

Here is an example:

← tet



dox42 Service Call POST

Parameters Settings Code View Testing About

The Domain Name Of Your Configured Dox42 Server *

demo.dox42.online

Query Portion Of The Desired Dox42 Service Call *

Operation=GenerateDocument&DocTemplate=https%3A%2F%2F[redacted].est.sharepoint.com%2FDynamics365%2FFor_Sales_Templates%2FSales_Quote_Flow.docx&InputParam.WhichQuote=Angebot&SharepointAction.Site=https%3A%2F%2F[redacted].est.sharepoint.com%2FDynamics365%2F&SharepointAction.Library=SalesQuotes&SharepointAction.FileName=TEEESEStQuote_Name.pdf_Dataverse_Flow.pdf&SharepointAction.Azure_AD=true&SharepointAction.Field1.Name=QuoteID&SharepointAction.Field1.Value=Angebots-ID&

Advanced parameters

Showing 2 of 2 Show all Clear all

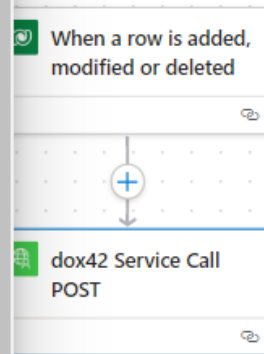
Request Body

```
{
  "UserId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
  "Ids": [
    {
      "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
    }
  ],
  "Environment": "https://exampleCompany.crm.dynamics.com",
  "CustomParameter1": "value",
  "CustomParameter2": "value"
}
```

Basic Accept Header, Leave As Is!

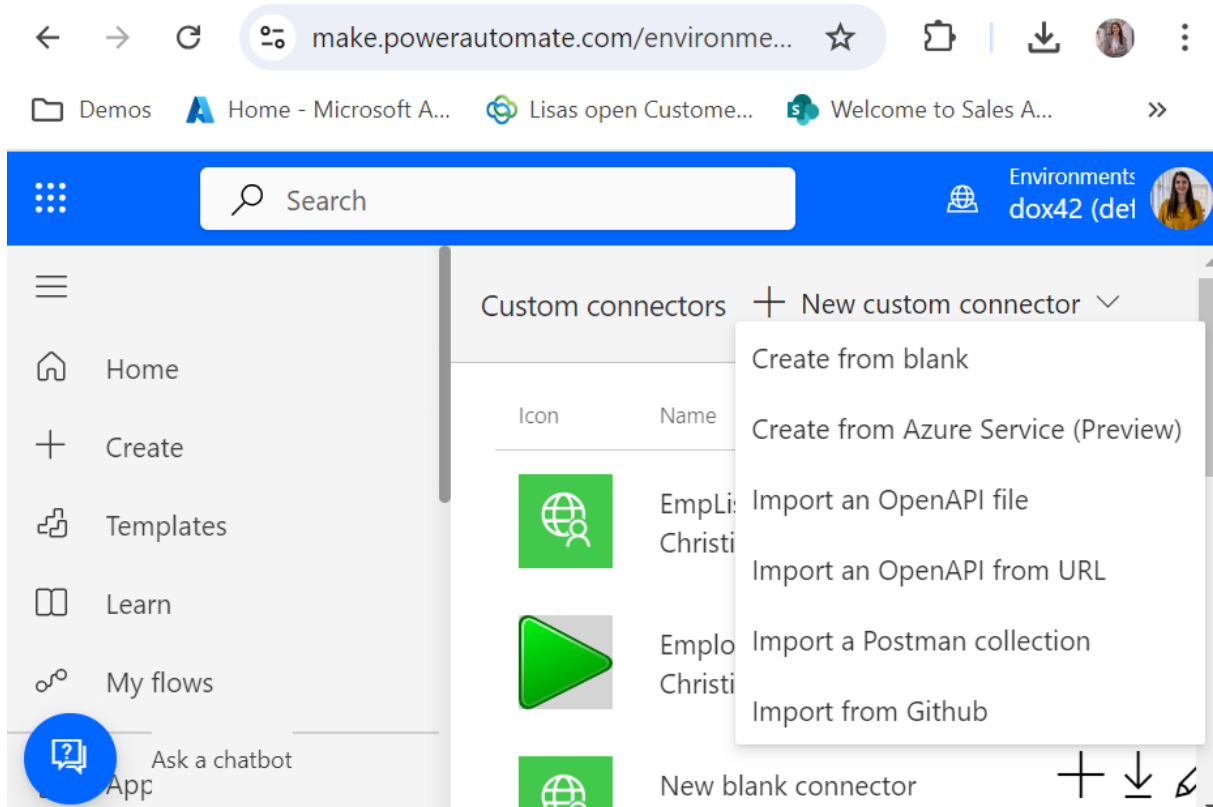
application/json

Connected to lisa.pulsinger@dox42.com. [Change connection](#)




3 Using the Custom Connector Method 2 (create from scratch)

As an alternative to the dox42 Custom Connector which you import via github, you can create a custom connector from scratch yourself:



Create from blank, and enter the URL of your dox42 Server or dox42 Online tenant:

General information



Upload connector icon
Supported file formats are PNG and JPG. (< 1MB)

↑ Upload

Icon background color

A color to show behind the icon (e.g., '#007ee5')

Description

Trigger dox42 Online from MS Flow

Connect via on-premises data gateway [Learn more](#)

Scheme *

HTTPS HTTP

Host *

yourtenant.dox42.online

Base URL

/

Security →

3.1.1 Security

In the Security Tab select "OAuth 2.0" and "Azure Active Directory"

Security

Choose the authentication type and fill in the required fields to set the security for your custom connector. [Learn more](#)

Authentication type

Choose what authentication is implemented by your API *

OAuth 2.0

OAuth 2.0

Identity Provider

Azure Active Directory

Enable Service Principal support

Client ID *

915cc84b-12c4-4196-a7ea-xxxxx

Client secret *

.....

Authorization URL

https://login.microsoftonline.com

Tenant ID

common

Resource URL *

https://xxxx.sharepoint.com

Enable on-behalf-of login

false

Scope

https://xxxx.sharepoint.com/AllSites.Write

Redirect URL

https://global.consent.azure-apim.net/redirect/new-20blank-20connector-5f48af0b749eb... 📄

← General
Definition →

Enter your App ID of your dox42 (Online) Server App registration from Azure Active Directory (see also dox42 AAD documentations), the Client Secret and the Resource URL of the SharePoint Online Tenant. For the Scope please enter <https://YourTenant.SharePoint.com/AllSites.Write> or <https://YourTenant.SharePoint.com/AllSites.Read> for the tenant where you store your templates.

In the last field "Redirect URL" you receive a custom redirect URL for each dox42 Custom connector. Copy it and add it as a Web Redirect URL to your dox42 (Online) Server App registration in Azure Active Directory AAD Admin Center (next to all other Web Redirect URIs that you've configured for dox42 Online:

Search Got feedback?

- Overview
- Quickstart
- Integration assistant
- Diagnose and solve problems
- Manage
 - Branding & properties
 - Authentication**

Web Quickstart Docs

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. [Learn more about Redirect URIs and their restrictions](#)

⚠ This app has implicit grant settings enabled. If you are using any of these URIs in a SPA with MSAL.js 2.0, you should migrate URIs. →

https://global.consent.azure-apim.net/redirect/dox42-20-2d-201-5f48af0b749eb93eb9-5f719450877f2d9fe5-1	
https://global.consent.azure-apim.net/redirect/dox42-20-2d-201-5f48af0b749eb93eb9-5f719450877f2d9fe5	

3.1.2 Definition and Test

To configure the dox42 Call in the Definition tab. Please create a dox42 REST link using the dox42 Server Designer and import it "Import from sample" in the Definition tab.

The MS Power Automate Connector will automatically recognize the parameters. You may define default values for these parameters. We would recommend to do so, as it will make testing a lot more convenient.

The screenshot shows the dox42 Server Designer interface. The main window is titled 'Documentation_PowerAutomate' and has tabs for '1. General', '2. Security', '3. Definition', and '4. Test'. The 'Definition' tab is active, showing a 'Request' section with fields for Verb (GET selected), URL (https://dox42test.dox42.online/dox42RestService.ashx), Path, Query, Headers, and Body. An 'Import from sample' dialog box is open on the right, with the same URL entered in the 'URL *' field. The dialog also has 'Import' and 'Close' buttons.

1. General > 2. Security > 3. Definition > 4. Test Swagger Editor ✓ Create connector ✗ Cancel

References (0)
References are reusable parameters used by both actions and triggers.

Policies (0)
Policies are used to change the behavior of actions and triggers through configuration. You can use one or more policies from a set of predefined templates.

[+ New policy](#)

Request + Import from sample

Verb *
The verb describes the operations available on a single path.

GET

URL *
This is the request URL.

Path
Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

Query
Query parameters are appended to the URL. For example, in /items?id=###, the query parameter is id.

Operation ... DocTemplate ... InputParam.Employ... ... SharepointAction.S... ...
SharepointAction.Li... ... SharepointAction.Fi... ... SharepointAction.A... ...

Headers
These are custom headers that are part of the request.

Accept ...

Body
The body is the payload that's appended to the HTTP request. There can only be one body parameter.

In the Test tab you can create a connection and test your connector.

3.1.3 Sharing is caring

Select your custom connector in the custom connectors list and share it with the people of your organization who should be able to use it. If you don't do so, nobody but yourself will be able to use your connector.

3.1.4 Your first MS Flow or Logic App with dox42

You can now create MS Flows using your custom connector.

The screenshot displays a Microsoft Flow Designer interface. At the top, a trigger step is configured with the following details:

- Trigger: When an item is created or modified
- Site Address: Employees - <https://dox42test.sharepoint.com/Employees>
- List Name: Employee List
- Show advanced options: expanded

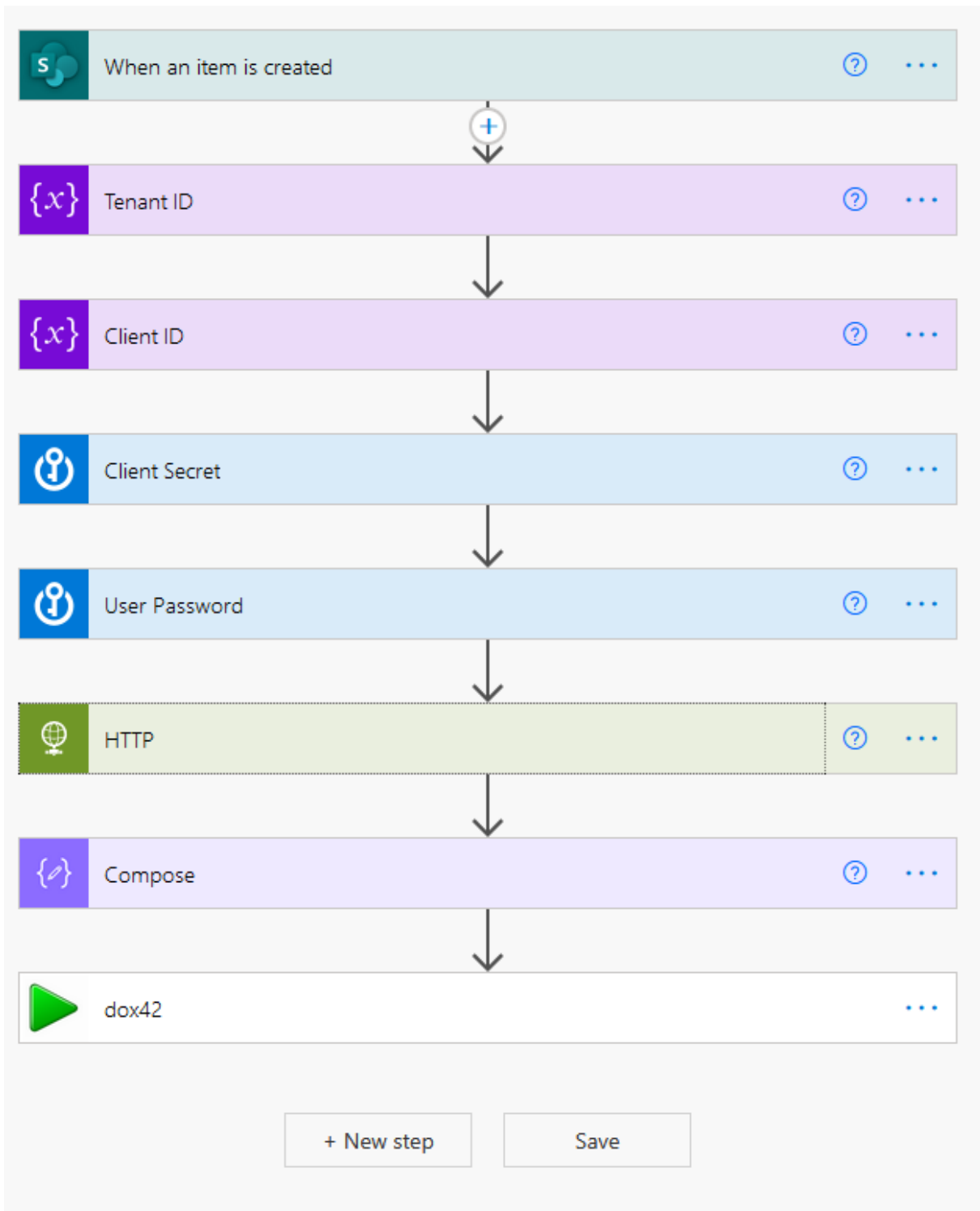
An arrow points down to the second step, a custom connector action named "call42":

- Operation: GenerateDocument
- DocTemplate: <https://dox42test.sharepoint.com/Employees/dox42Templates/EmployeeList.docx>
- SharepointAction.Library: TargetLib
- SharepointAction.Site: <https://dox42test.sharepoint.com/Employees>
- SharepointAction.FileName: EmployeeListFlow.pdf
- SharepointAction.azure_ad: Yes
- Accept: application/json

At the bottom of the flow editor, there are two buttons: "+ New step" and "Save".

4 Using the dox42 Premium Connector, Method 1

We will build a flow that generates a bearer access token and executes a dox42 service call.



4.1 Generating the ID Token

In order to generate a valid access or ID token, we have to send a request to Microsoft's token API (<https://login.microsoftonline.com/<Tenant-ID>/oauth2/v2.0/token>). In this example we will be using a client secret and username and password to get an ID token, since we will be working with SharePoint, which does not accept access tokens generated with a client secret. The parameters needed for this are as follows:

grant_type	Should be password
client_id	Your Client ID
client_secret	Client secret created in Azure AD
scope	For example, a SharePoint scope: yourcompany.sharepoint.com/.default
username	Microsoft username
password	Microsoft password

To get a client secret go to your dox42 App you have registered beforehand in Azure Active Directory admin center. Go to **Certificate & secrets** → **Client secrets** → **New client secret**

To get the client ID and tenant ID go to the **Azure AD overview** of your application. You will find the application ID (client ID) and tenant ID there.

The screenshot shows the Azure AD application overview for 'dox42 Online'. The left sidebar contains navigation options: Overview (selected), Quickstart, Integration assistant, Manage (Branding, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles), and a search bar. The main content area shows an 'Essentials' section with the following details:

- Display name: dox42 Online
- Application (client) ID: [Redacted]
- Object ID: [Redacted]
- Directory (tenant) ID: [Redacted]
- Supported account types: My organization only

There is a warning banner at the top: 'A certificate or secret is expiring soon. Create a new one →'. At the bottom, there are links for 'Get Started' and 'Documentation'. A blue information banner at the bottom states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authent'.

We highly recommend using Azure Key Vault or similar services to encrypt secrets and passwords in your flow.

4.2 Setting up an Azure Key Vault

To create an Azure key vault, you can refer to these articles:

<https://blog.pragmaticworks.com/azure-active-directory-and-resource-groups>

<https://azuredevopslabs.com/labs/vstsextend/azurekeyvault/> (Task 2)

Once you have created an Azure key vault navigate to “Secrets” and Generate/Import a new secret

Create a secret ...

Upload options	Manual
Name *	examplePassword
Value *
Content type (optional)	
Set activation date	<input type="checkbox"/>
Set expiration date	<input type="checkbox"/>
Enabled	<input checked="" type="radio"/> Yes <input type="radio"/> No
Tags	0 tags

Simply enter the credentials and press “Create” (bottom left of window).

dox42PowerAutomate | Secrets ...

Key vault

Search (Ctrl+/) << + Generate/Import Refresh Restore Backup Manage deleted secrets

Name	Type
ClientAssertion	
clientSecret	
password	
username	

4.3 Configuring the Flow

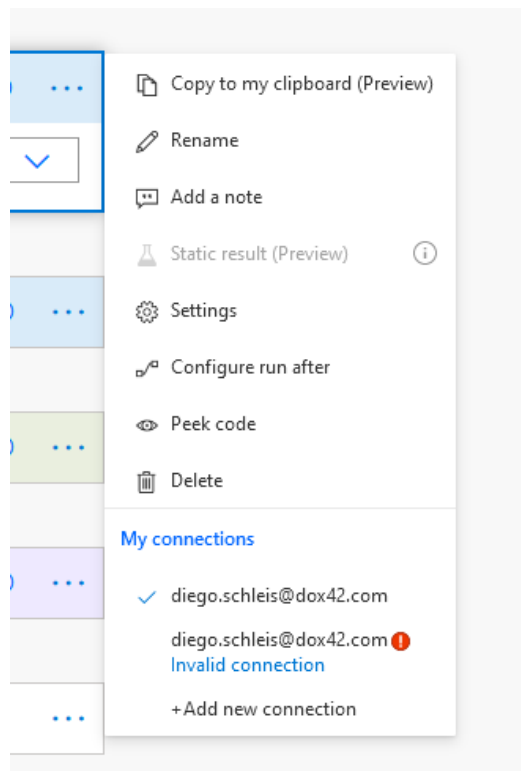
Now that we have the necessary HTTP request parameters with values, we can start to implement them in the flow. You have already seen the finished flow in the first illustration of this documentation, now we will look at each component in detail.

First of all, we need a trigger to start the flow. Any trigger will suffice, in the example we use a simple SharePoint trigger that triggers the flow as soon as a file is created in the target Folder.

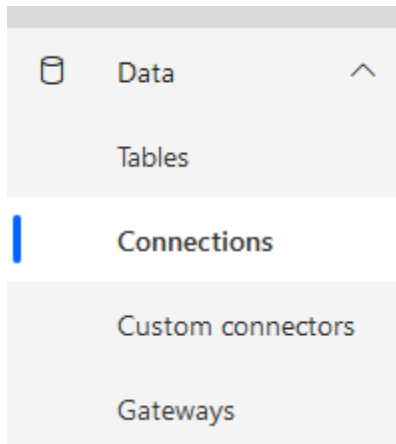
Next, we highly recommend using variables in your flow. In this example client ID and tenant ID are saved as variables, since they are not as sensitive as other credentials, but you can also save those in your Azure key vault.

The screenshot shows two variable definition cards in the Power Automate interface. The top card is for 'Tenant ID' with a name of 'tenantId', type of 'String', and a redacted value. The bottom card is for 'Client ID' with a name of 'clientId', type of 'String', and a redacted value. Both cards have a purple header with a variable icon and a question mark icon.

Next, we need to retrieve the credentials from the Azure key vault. To achieve this, add a new Action then search for **Azure Key Vault** and choose **Get secret**.



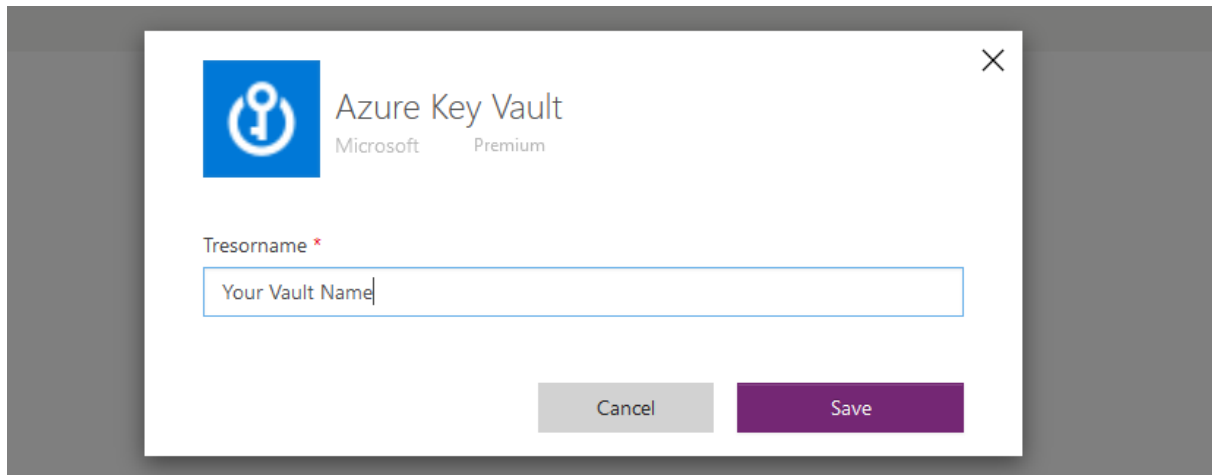
If you get an invalid connection as soon as you add the Azure key vault action, open the **Connections** list in a new browser tab. You can find said list here:



Next open the faulty connection and do the steps illustrated below:



A screenshot of the Microsoft Power Platform 'Connections' page for a faulty Azure Key Vault connection. The page shows the following details:

- Connector name:** Azure Key Vault
- Description:** Azure Key Vault ist ein Dienst zum sicheren Speichern und Zugreifen auf Geheimnisse.
- Status:** Parameter value missing. [Update password](#)
- Owner:** Diego Schleis
- Created:** 20.8.2021, 08:38:36
- Modified:** 20.8.2021, 08:38:38

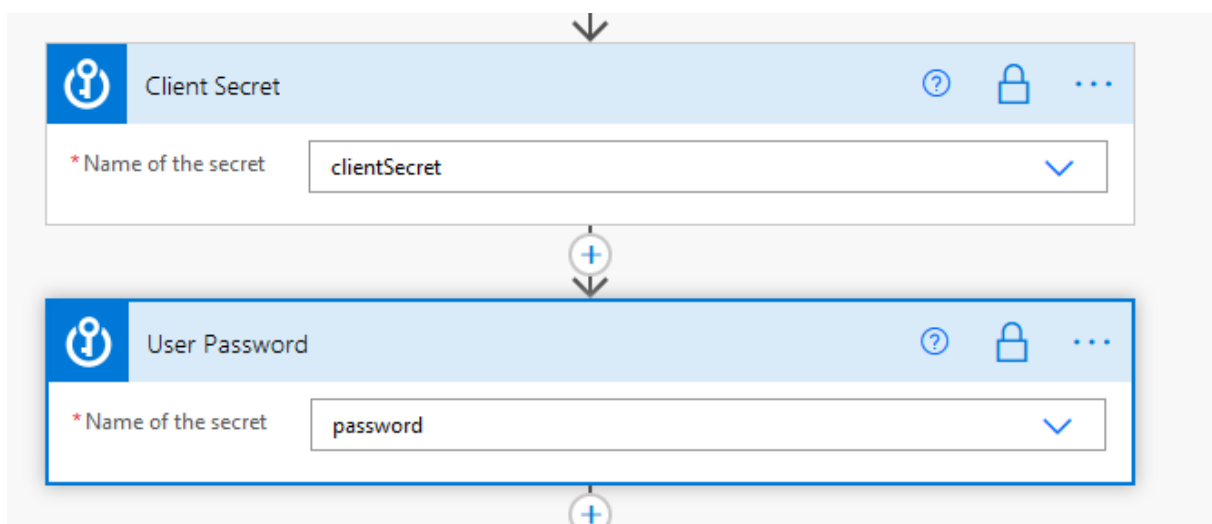


Clicking **Update password** will pop-up a Microsoft authentication window.

After that the connection should be successful.

Name	Modified	Status
 diego.schleis@dox42.com SharePoint	... 2 h ago	Connected
 diego.schleis@dox42.com Azure Key Vault	... 3 min ago	Connected

In the flow, click the connection again to get it working.



It is highly recommended to enable the **Secure Outputs** option. To do this click the three dots and click settings, there you can enable secure outputs. This will hide the secrets in flow runs.

Next we need to set up the HTTP request to generate an ID token. For that create a HTTP action:

The screenshot shows the configuration for an HTTP action. The fields are as follows:

- Method:** POST
- URI:** `https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token`
- Headers:**
 - Content-Type: `application/x-www-form-urlencoded`
 - Enter key: Enter value
- Queries:** Enter key: Enter value
- Body:** `grant_type=password&client_id={clientId}&client_secret={secret}&scope=user.read openid profile offline_access&username=diego.schleis@dox42.com&password={password}`
- Cookie:** Enter HTTP cookie

A link for "Show advanced options" is located at the bottom left of the configuration panel.

As you can see, the variables and secrets from the key vault are used for the HTTP request. Additionally, you have to define the **content-type** in the header with the value **application/x-www-form-urlencoded**.

We recommend also enabling secure outputs in the HTTP action to hide the access token.

Next, create a **Compose** action and enter the following expression: `outputs('HTTP').body?['id_token']`

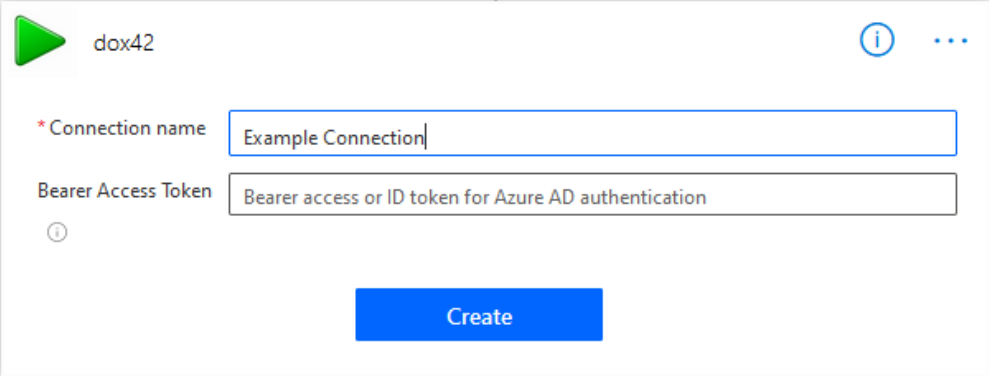
The screenshot shows the configuration of a Compose action in a workflow editor. The input field contains the expression `outputs('HTTP').body?['id_token']`. A tooltip on the right provides instructions on dynamic content and shows the same expression.

Dynamic content Expression

```
fx outputs('HTTP').body?['id_token']
```

OK

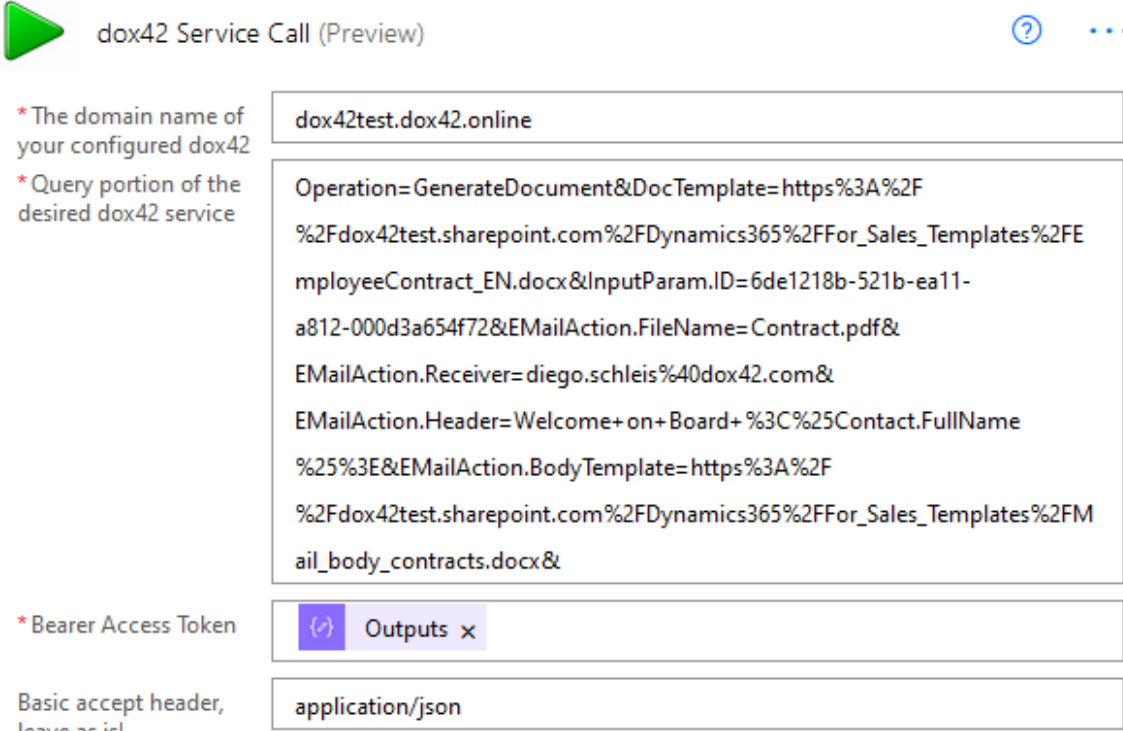
Finally, we will configure the dox42 premium connector. **An important piece of information to know before getting started** is that the dox42 premium connector relies on external means of authentication, but still has to have an authentication type according to the connector certification rules. That means that the connection to the connector is obsolete, and the **Bearer Access Token** parameter shall be left empty, since it is overwritten anyways. So, build your connection like this:



dox42

* Connection name

Bearer Access Token



dox42 Service Call (Preview)

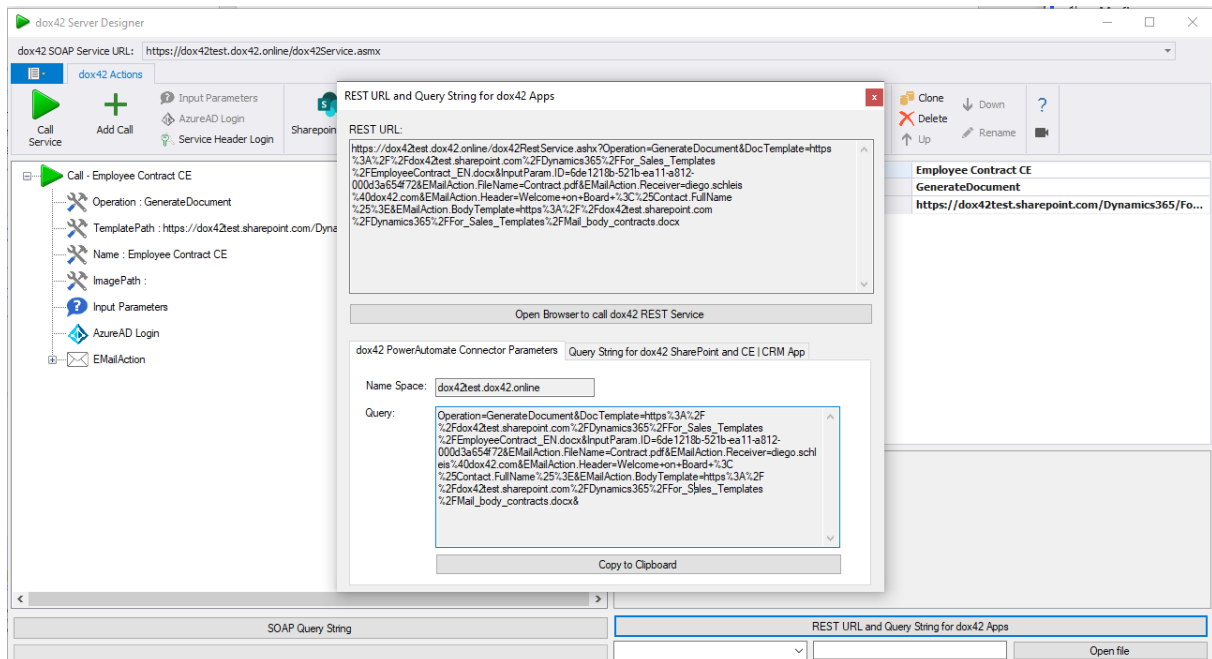
* The domain name of your configured dox42

* Query portion of the desired dox42 service

* Bearer Access Token

Basic accept header, leave as is!

Both “domainname” and “querystring” can be retrieved from the dox42 Server Designer (Version 1.0.1.4 and later) like this:

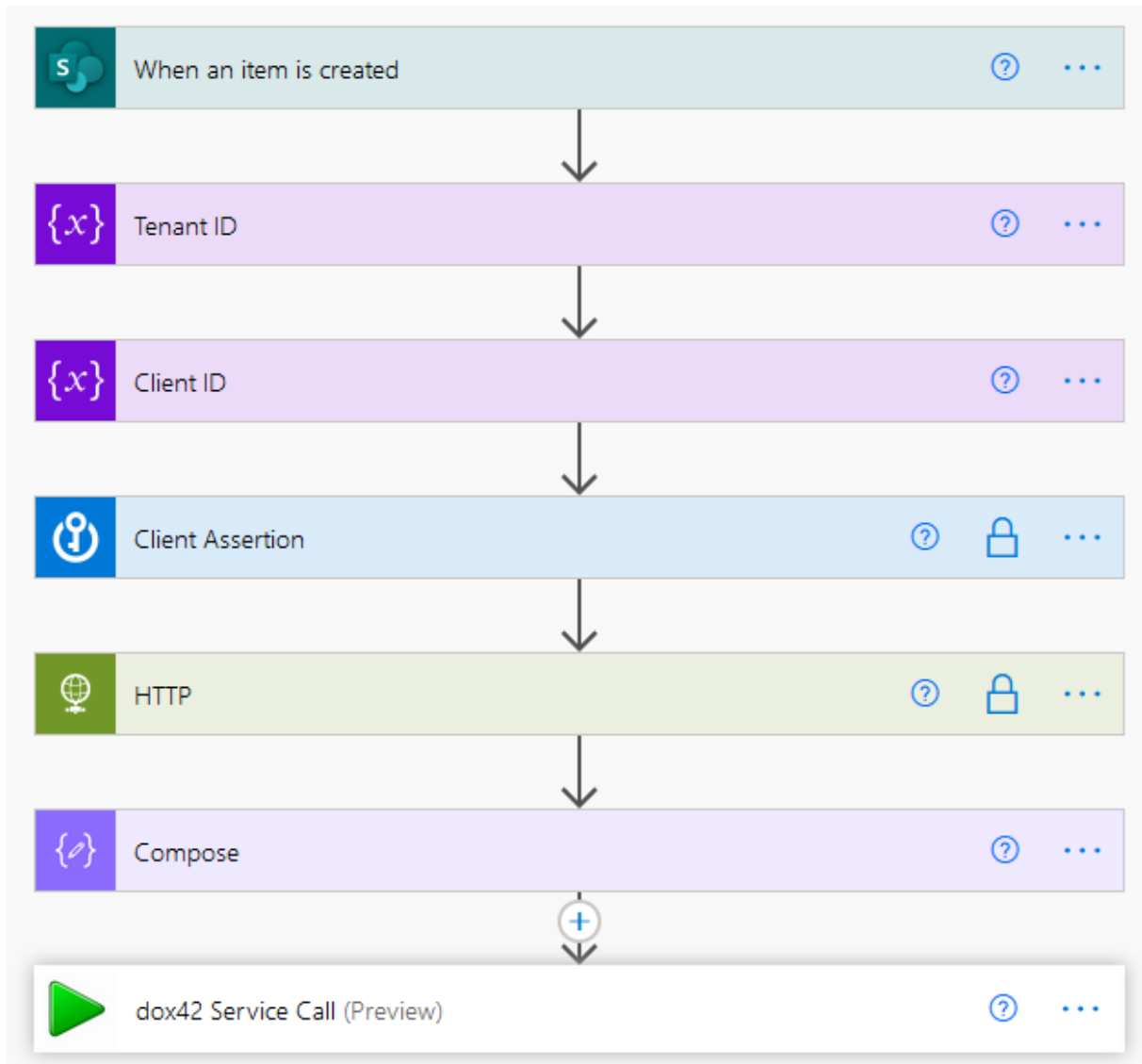


Configure a working config in the server designer and navigate to “REST URL and Query String for dox42 Apps” in the “dox42 Power Automate Connector Parameters” you will find the correct parameters for the dox42 connector. “Name Space” represents the “domainname” parameter and “Query” the “querystring” parameter.

For the **Bearer Access Token** parameter, use the Outputs of the compose action.

Now you have a proper setup to run dox42 server calls in your Power Automate flows with the new dox42 premium connector.

5 Using the dox42 Premium Connector, Method 2



If you wish to work with access tokens instead of ID tokens and also need access to a SharePoint, you cannot use a client secret for authentication. The correct way to achieve this is via certificate.

Method 2 will describe the process of uploading certificates to Azure AD and retrieving a client assertion for the token generation. If, however, you do not work with SharePoint but rather with other services that accept client secrets as means of authentication you can replace the **client_assertion** parameter shown later with **client_secret**.

5.1 Generating the Access Token

In order to generate a valid access or ID token, we have to send a request to Microsoft's token API (<https://login.microsoftonline.com/<Tenant-ID>/oauth2/v2.0/token>). This method will use a certificate for the Azure AD access token generation. The parameters needed for this are as follows:

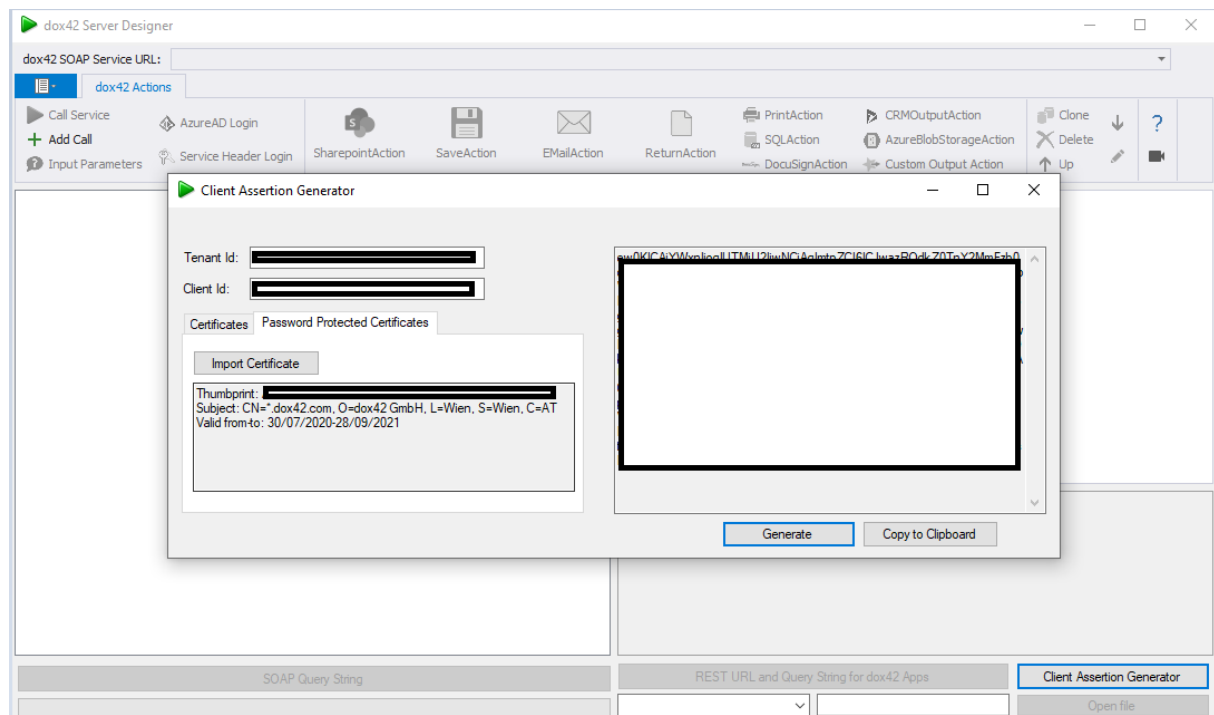
grant_type	Should be client_credentials
client_id	Your Client ID
client_assertion	A JWT token signed with a certificate
scope	For example, a SharePoint scope: yourcompany.sharepoint.com/.default
client_assertion_type	Should be urn:ietf:params:oauth:client-assertion-type:jwt-bearer

To get the client ID and tenant ID refer to method 1.

Firstly, upload the certificate file you want to use for generating the access token to Azure AD. To Achieve this, simply navigate to **"Certificates & secrets" in your Azure AD app** and upload a new certificate there.

Now, generating a client assertion is quite tricky, therefore **Server Designer version 1.0.1.6** or later offers the possibility to generate client assertions from normal and password protected certificate files.

After that, navigate to "Client Assertion Generator" and import your public and private key certificate files, or complete certificate file, alongside your tenant and client ID and press generate. Now you have a working client assertion that you can save to Azure key vault as explained in method one.

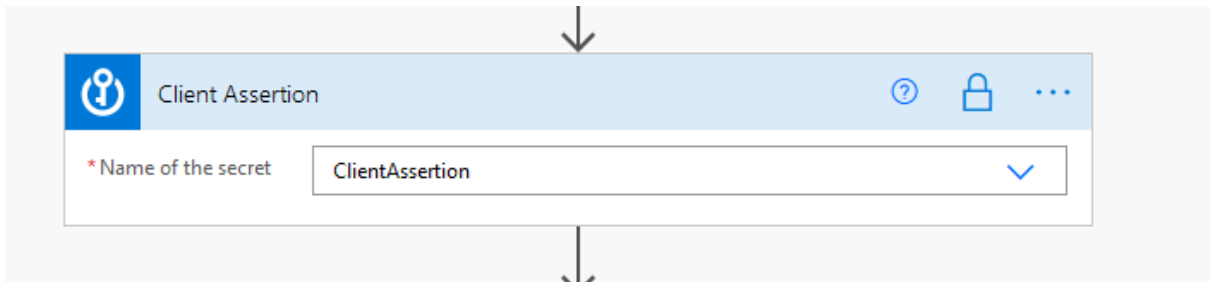


If you wish to do this step without the Server Designer you can refer to this article especially: <https://docs.microsoft.com/en-us/azure/architecture/multitenant-identity/client-assertion>

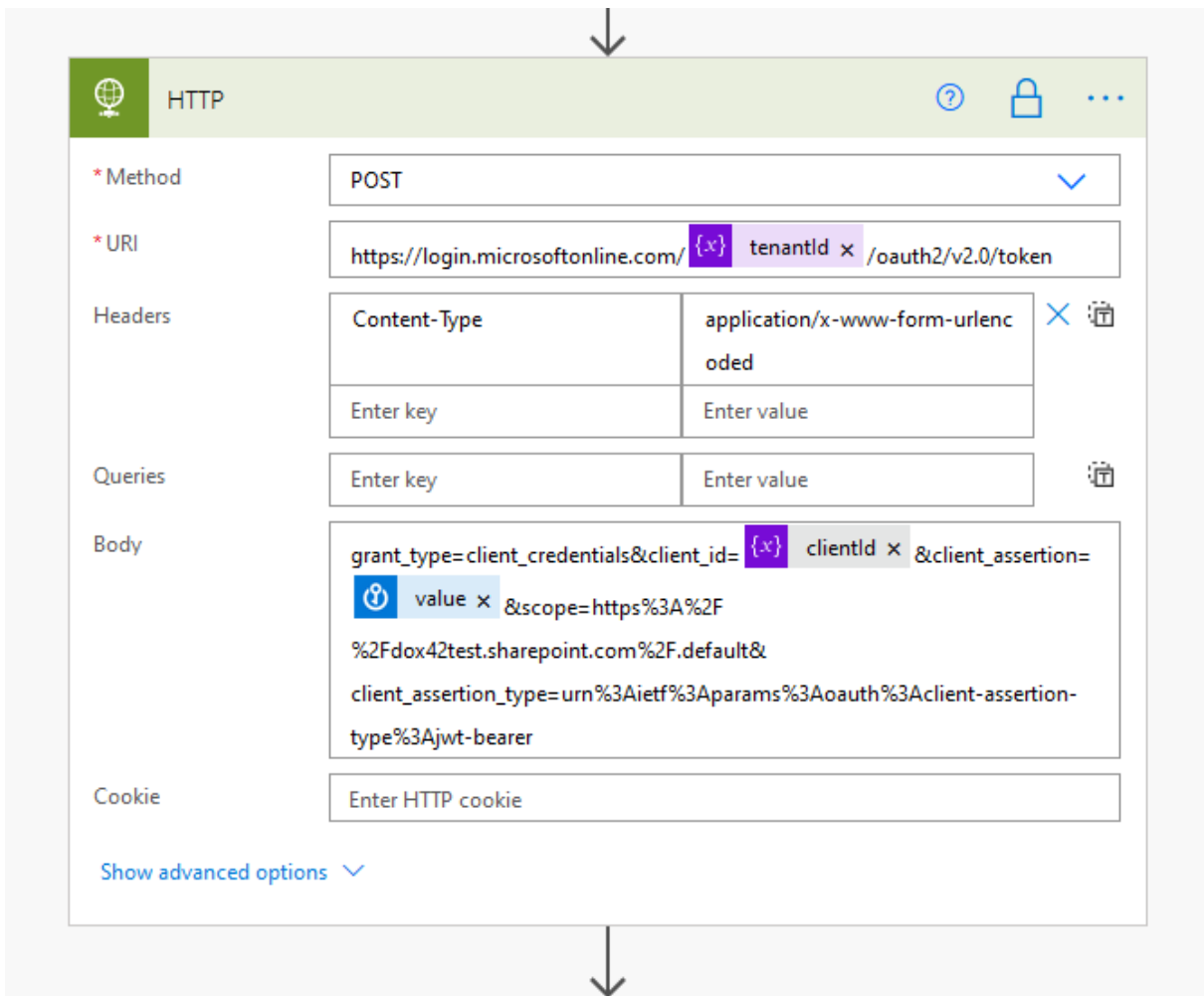
Again, we highly recommend using Azure Key Vault or similar services to encrypt secrets and passwords in your flow.

5.2 Configuring the Flow

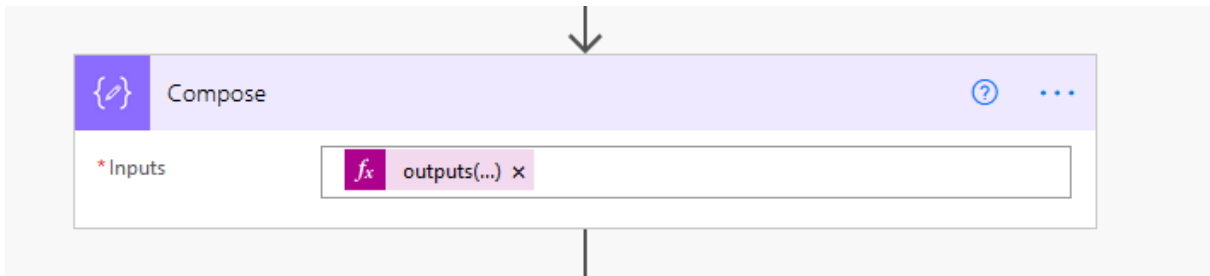
For the majority of the steps, especially Azure key vault, you can refer to method one.



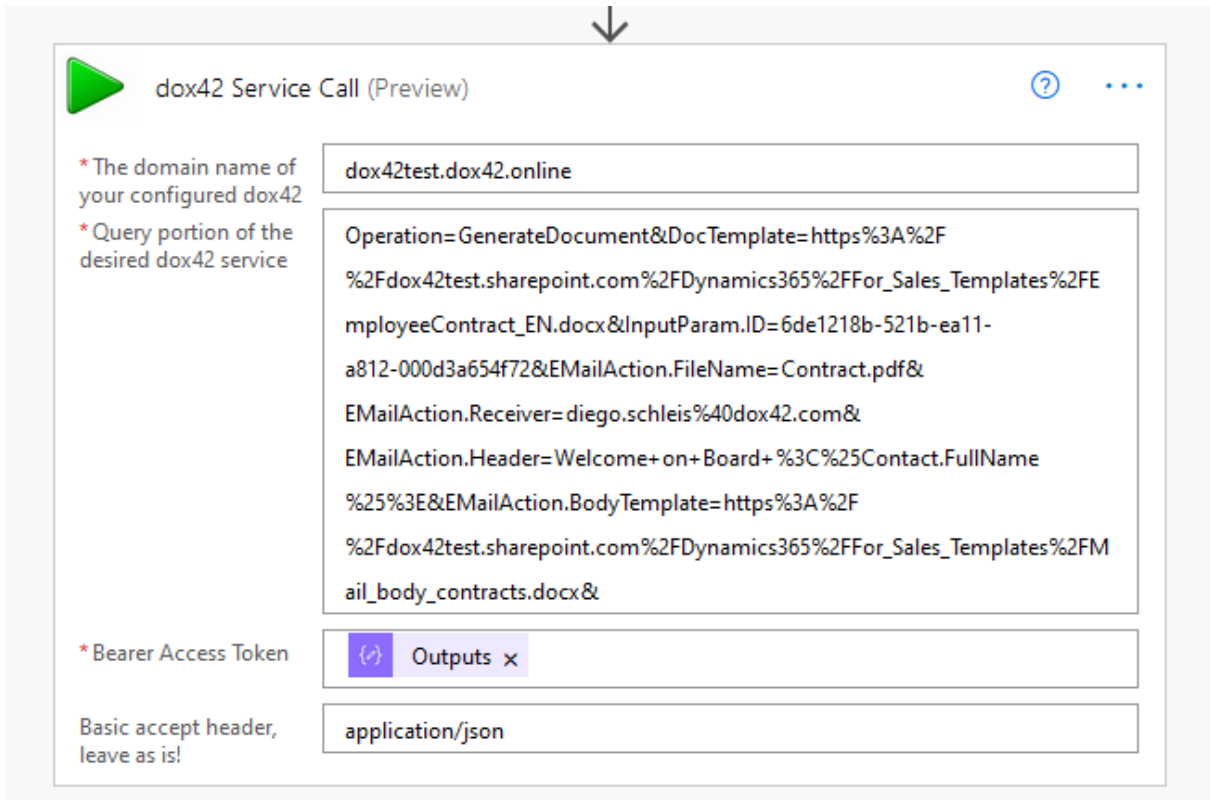
Get the client assertion from Azure key vault.



Again, the variables and secrets from the key vault are used for the HTTP request. Additionally, you have to define the **content-type** in the header with the value **application/x-www-form-urlencoded**.



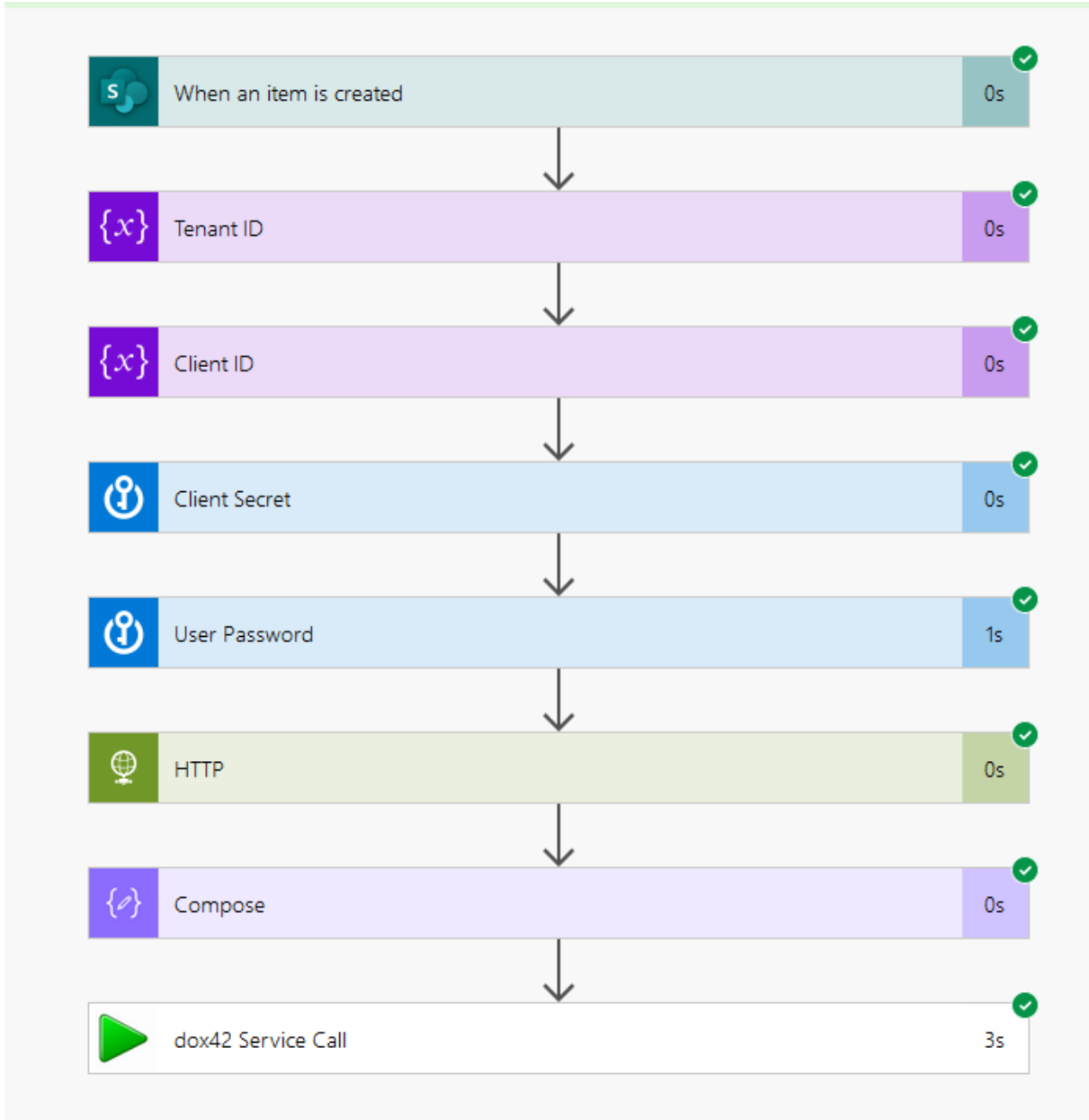
To retrieve the access token from the HTTP call, enter this expression:
outputs('HTTP').body?['access_token']



The dox42 service call action is the same for this method.

6 Running the Flow

To run the flow we just created, save the flow, and click test. Next, you have to manually trigger the flow the first time. In the case of the flow example in the screenshot below an item has to be created in the target SharePoint list to trigger it. After doing that, the flow will start and will generate your desired document for you.



Congratulations, you have built your first flow using the dox42 premium connector!

Good luck and may the Flow be with you.