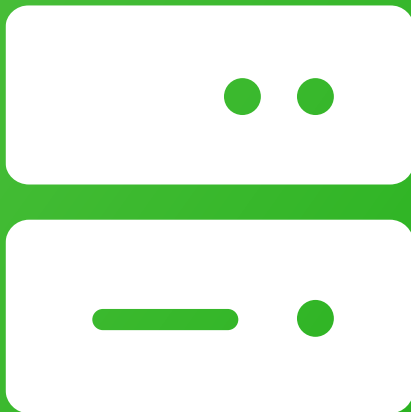


POWER AUTOMATE CONNECTOR

DOCUMENTATION

Version 2.2



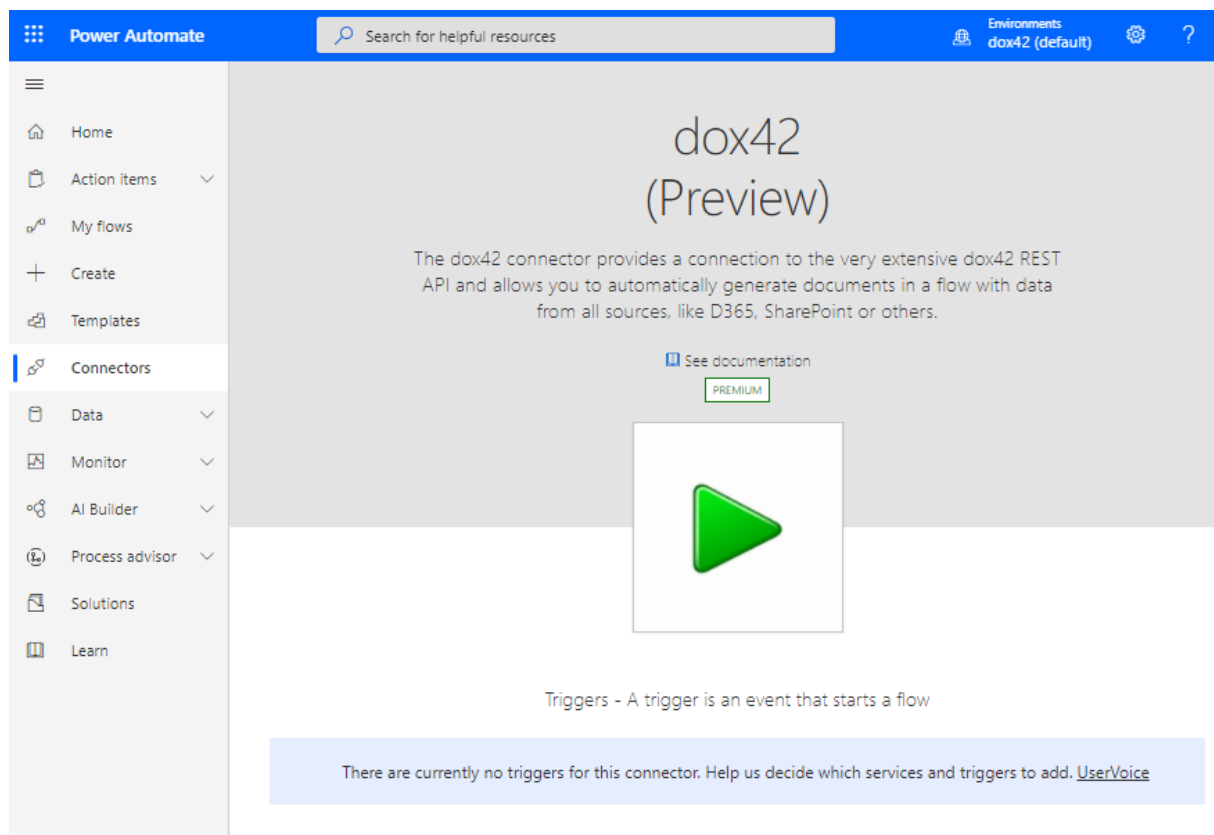
Contents

1	Introduction	3
2	Using the Custom Connector Method 1 (create from scratch)	4
2.1	Security	5
2.2	Definition and Test	7
2.2.1	Create GET action.....	7
2.2.2	Setup Request section	9
2.2.3	Create a POST Action.....	11
2.3	Create a Policy	12
2.4	Sharing is caring	13
2.5	Your first MS Flow or Logic App with dox42	13
3	Using the dox42 Custom Connector Method 2 (import via Github)	14
3.1	Security	15
3.2	Definition and Test – Check Policy	17
3.3	Sharing is Caring	18
3.4	dox42 Custom Connector Example Flow	19
3.4.1	dox42 Custom Connector Example Flow with POST Call	19
4	Using the dox42 Premium Connector, Method 1	21
4.1	Generating the ID Token	22
4.2	Setting up an Azure Key Vault	22
4.3	Configuring the Flow	23
5	Using the dox42 Premium Connector, Method 2	30
5.1	Generating the Access Token	31
5.2	Configuring the Flow	32
6	Running the Flow	34

1 Introduction

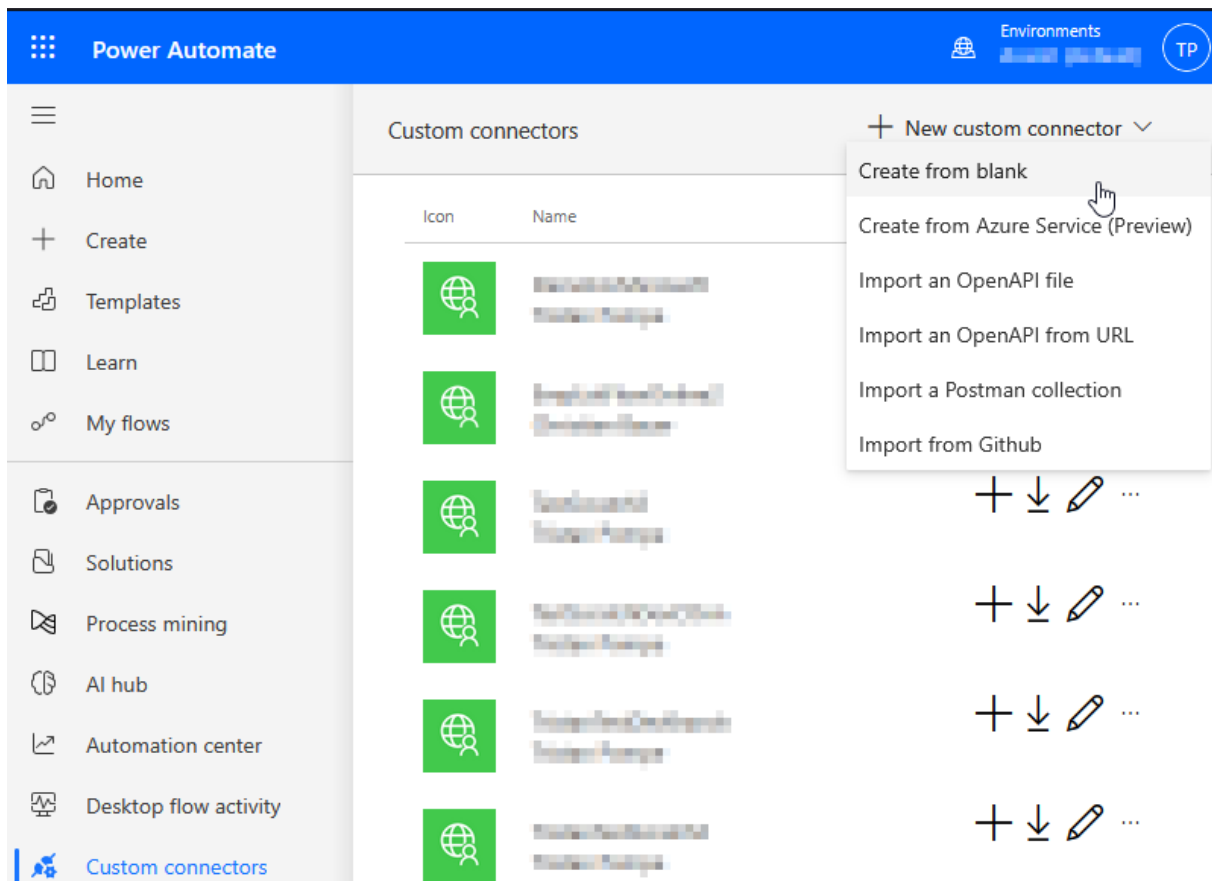
The dox42 Power Automate Premium Connector provides a connection to the very extensive dox42 REST API and allows you to automatically generate documents in a flow with data from all sources like D365, SharePoint or others. You can also use it from Azure Logic Apps and Power Apps. The connector is available 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.




2 Using the Custom Connector Method 1 (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, choose a name and enter the URL of your dox42 Server or dox42 Online tenant:

General information



Upload

Upload connector icon

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

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://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 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

2.2.1 Create GET action

Power Automate

Connector Name: DokuConnector

1. General > 2. Security > 3. Definition > 4. AI Plugin (preview) > 5. Code > 6. Test

Swagger editor | Create connector | Cancel

Actions (0)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.
+ New action

Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.
+ New trigger

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

Start by adding an action or trigger on the left.

Security | AI Plugin (preview)

First, you need to create a new action.

✓ Actions (1)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

!

1

Dox42Call

...

+

New action

✓ Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

+

New trigger

✓ 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

General

Summary [Learn more](#)

1 dox42 Service Call GET

Description [Learn more](#)

2 Make an HTTP GET request to the dox42 service

Operation ID *
This is the unique string used to identify the operation.

3 Dox42Call

Visibility [Learn more](#)

☒ none ☐ advanced ☐ internal ☐ important

Request
It defines the pre-requirements needed in order to make a request. Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

4 + Import from sample

Response
It defines the shape of response returned by the underlying connector when making the request.


default

 default

+ Add default response

Fill the necessary input boxes:

1. Summary: dox42 Service Call GET
2. Description: Make an HTTP GET request to the dox42 service
3. OperationID: Dox42Call

Page 8 (of 34) 

2.2.2 Setup Request Section

Now you need to setup the Request section. Therefore click on "Import from sample". As Verb select GET and for the URL you can use "/dox42RestService.ashx?querystring" and click import.

The screenshot shows the dox42 configuration interface. On the left, there are sections for Actions (1), Triggers (0), References (0), and Policies (0). The main area is divided into General, Request, and Response sections. In the General section, the Summary is "dox42 Service Call GET", the Description is "Make an HTTP GET request to the dox42 service", the Operation ID is "Dox42Call", and the Visibility is set to "none". In the Request section, the "Import from sample" button is highlighted with a green arrow and a circled '1'. In the Response section, the "default" response is selected. On the right, the Verb is set to GET (circled '2'), the URL is "/dox42RestService.ashx?querystring" (circled '3'), and the Headers are "Content-Type application/json" and "Accept application/json". At the bottom right, the "Import" button is highlighted with a green arrow and a circled '4', and the "Close" button is also visible.

Next, click on the dropdown arrow next to the querystring field and select edit:

The screenshot shows the Request section configuration. The Verb is set to GET, the URL is "https://dox42support.dox42.online/dox42RestService.ashx", and the Path is empty. The Query section shows a dropdown menu for "querystring" with a green arrow pointing to the dropdown arrow. The dropdown menu is open, showing "Edit" and "Delete" options. The Body section is empty.

Make the parameter required. After you have done that, you can go back. There is no Save button.

[← Back](#) **2**

Parameter

Name *

Description [Learn more](#)

Summary [Learn more](#)

Default value

Is required?
☒ Yes ☐ No

Visibility [Learn more](#)
☒ none ☐ advanced ☐ internal ☐ important

Location *
☐ Path ☒ Query ☐ Header ☐ Body

Type

Format

Dropdown type [Learn more](#)
☒ Disabled ☐ Static ☐ Dynamic

2.2.3 Create a POST Action

Follow the steps in this chapter, if you want to include a POST Call in your dox42 Power Automate Connector.

1. General > 2. Security > **3. Definition** > 4. AI Plugin (preview) > 5. Code > 6. Test

Swagger editor

Actions (2)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

- ✓ 1 Dox42Call
- ! 2 Dox42CallPost
- + New action

Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

- + New trigger

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

General

Summary [Learn more](#)

2 dox42 Service Call POST

Description [Learn more](#)

3 Make an HTTP POST request to the dox42 service

Operation ID *

This is the unique string used to identify the operation.

4 Dox42CallPost

Visibility [Learn more](#)

☒ none ☐ advanced ☐ internal ☐ important

Request

It defines the pre-requirements needed in order to make a request. Des A unique parameter is defined by a combination of a name and location

+ Import from sample

Import from sample

Verb *

☐ GET ☐ DELETE ☒ POST ☐ PUT ☐ HEAD ☐ PATCH

URL *

/dox42RestService.ashx?querystring

This is the request URL.

Headers

Headers separated by a new line, e.g.:
Content-Type application/json
Accept application/json

These are custom headers that are part of the request.

Body

{}

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

Import Close

1. Create Action
2. Summary: dox42 Service Call POST
3. Description: Make an HTTP POST request to the dox42 service
4. OperationID: Dox42CallPost
5. Choose POST verb
6. URL: /dox42RestService.ashx?querystring
7. Body: Just add {}
8. Import

Please make the querystring parameter required (see chapter Setup Request Section) for the querystring parameter.

2.3 Create a Policy

Actions (1)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

✓ 1 Dox42Call ...

+ New action

Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

+ New trigger

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

General

Summary [Learn more](#)

dox42 Service Call GET

Description [Learn more](#)

Make an HTTP GET request

Operation ID *

This is the unique string used

Dox42Call

Visibility [Learn more](#)

☒ none ☐ advanced

Request

It defines the pre-requisite. A unique parameter is defined.

Request

Verb *

The verb describes the operation

GET

URL *

This is the request URL.

[https://<yoururl>/dox42RestService.ashx?@queryParameters\('querystring'\)](#)

Actions (1)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

✓ 1 Dox42Call ...

+ New action

Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

+ New trigger

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

Policies (1)
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.

! 1 Define dox42 ...

+ New policy

Policy details

Name *

1 Define dox42 Service Url

Template * [Learn more](#)

Choose a template

Set property (Preview)

Set header/query parameter value from URL (Preview)

Set host URL 2

Set connection status to unauthenticated (Preview)

Route request

Convert delimited string into array of objects (Preview)

Convert an object to an array (Preview)

Set query string parameter

Set HTTP header

Convert an array to an object (Preview)

Set the name to “Define dox42 Service Url”.

The operation can remain blank.

In the URL Template box please enter

“https://<yoururl>/dox42RestService.ashx?@queryParameters('querystring')” and put your dox42 service URL in the placeholder. For example:

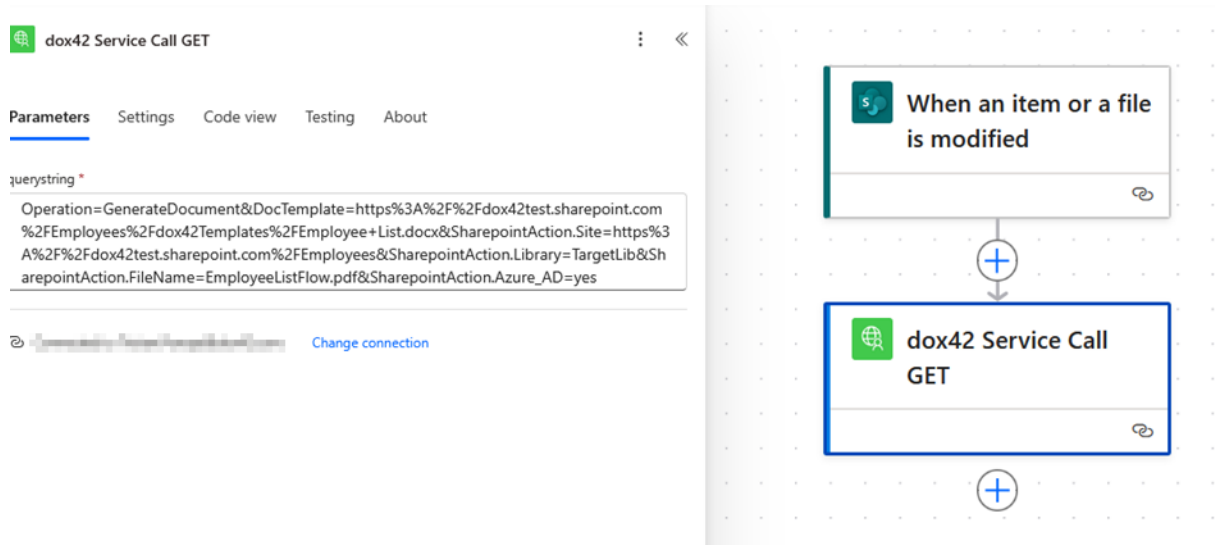
“https://dox42tenanttest.dox42.online/dox42RestService.ashx?@queryParameters('querystring')”

2.4 Sharing is caring

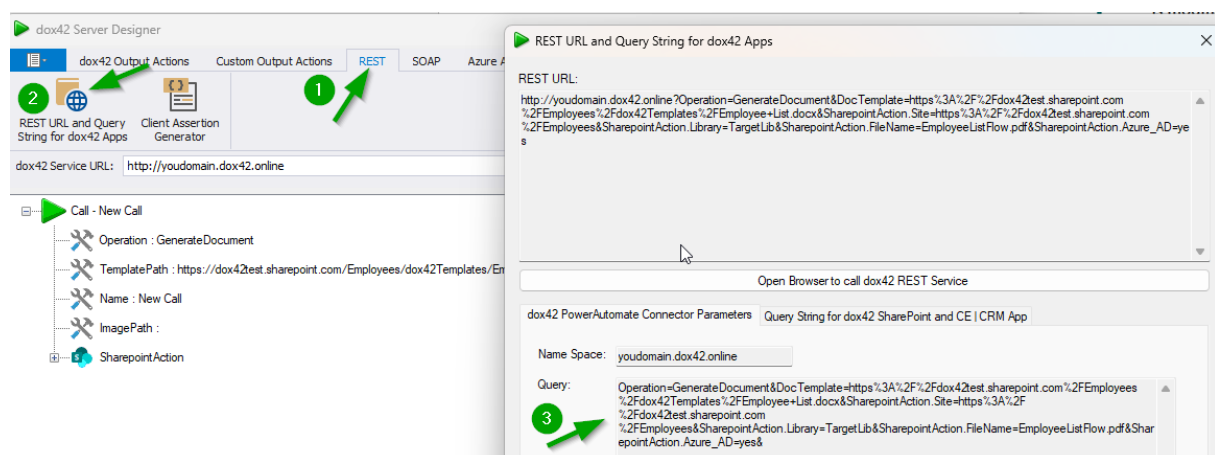
Select your custom connector in the custom connectors list and share it with the people of your organization who should be able to use it. If you don't do so, nobody but yourself will be able to use your connector.

2.5 Your first MS Flow or Logic App with dox42

You can now create MS Flows using your custom connector.



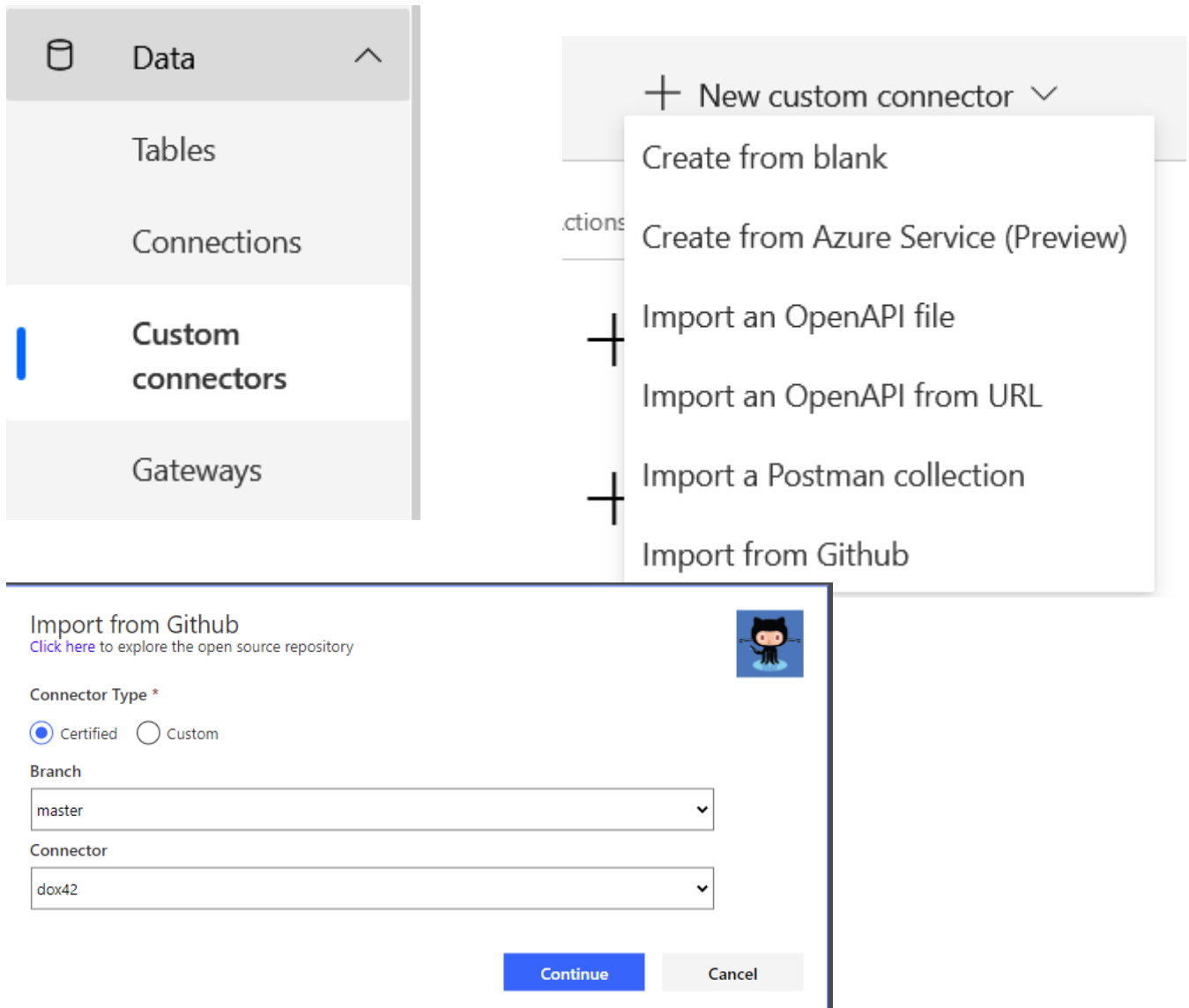
With the dox42 Server designer you can generate the querystring.



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

As an alternative to the set-up of the custom connector from scratch, you can generate your documents from Microsoft Power Automate Flows via our Github imported custom connector.


It can be found in “certified connectors” in the “master” branch:



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

Upload connector icon

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

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 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 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](#)

Warning: 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-201-5f48afb749eb93eb9-5f719450877f2d9fe5-1	
https://global.consent.azure-apim.net/redirect/dox42-20-2d-201-5f48afb749eb93eb9-5f719450877f2d9fe5	

3.2 Definition and Test – Check Policy

Request

It defines the pre-requirements needed in order to make a request. Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

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.

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

* domainname	* querystring	accept
--------------	---------------	--------

Edit

Delete

ended to the HTTP request. There can only be one body parameter.

Please delete all parameters in Headers.

Please repeat this step also for the "Dox42CallPost" action. Request body does not have to be deleted.

Actions (2)
Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

- 1 Dox42Call ...
- 2 Dox42CallPost ...
- + New action

Triggers (0)
Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

- + New trigger

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

Policies (1)
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.

- 1 Define dox42 ... 1
- + New policy

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.

Url Template *

Specifies template from which host URL will be generated.

2 https://<yourdox42Service>/dox42RestService.ashx?@headers('querysting')

← Security AI Plugin (preview) →

1. Go to the Policy “Define dox42...”

2. Change the “Url Template” to

`https://<yourdox42serviceurl>/dox42RestService.ashx?@queryParameters('querysting')`

and put your dox42 service URL in the placeholder. For example:

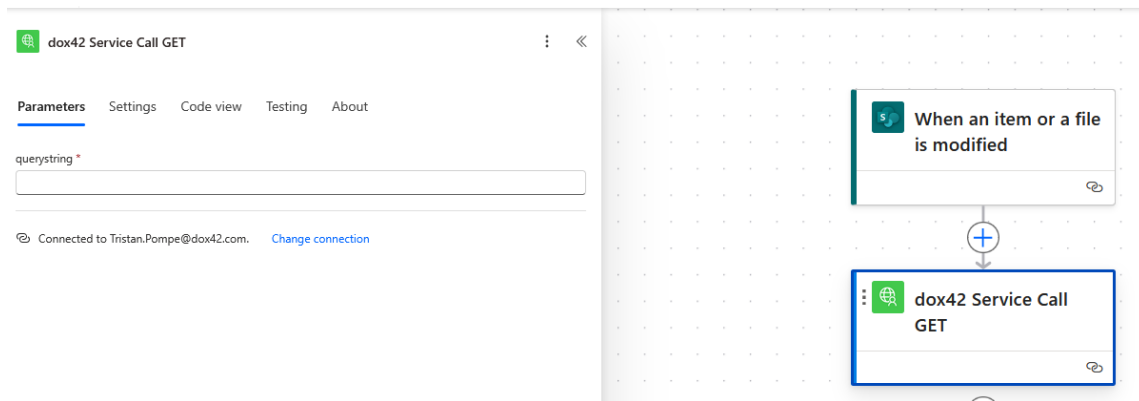
`“https://dox42tenanttest.dox42.online/dox42RestService.ashx?@queryParameters('querysting')”`

And follow the documentation in the chapter **2.2.2.2.2Setup Request Section**.

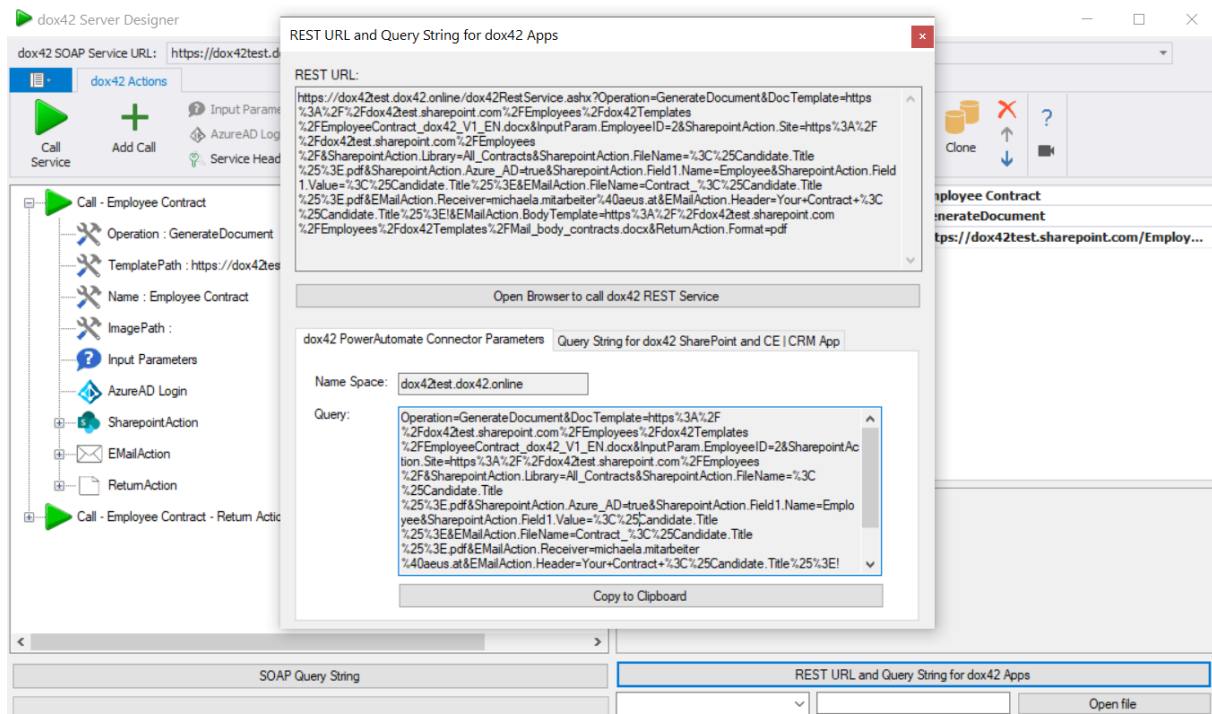
3.3 Sharing is Caring

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.

3.4dox42 Custom Connector Example Flow



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 “querystring” parameter. Just copy and paste it into your Flow.



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

3.4.1 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%2Fest.sharepoint.com%2FDynamics365%2FFor_Sales_Templates%2FSales_Quote_Flow.docx&InputParam.WhichQuote=

Angebot&SharepointAction.Site=https%3A%2F%2Fest.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](#)

When a row is added, modified or deleted

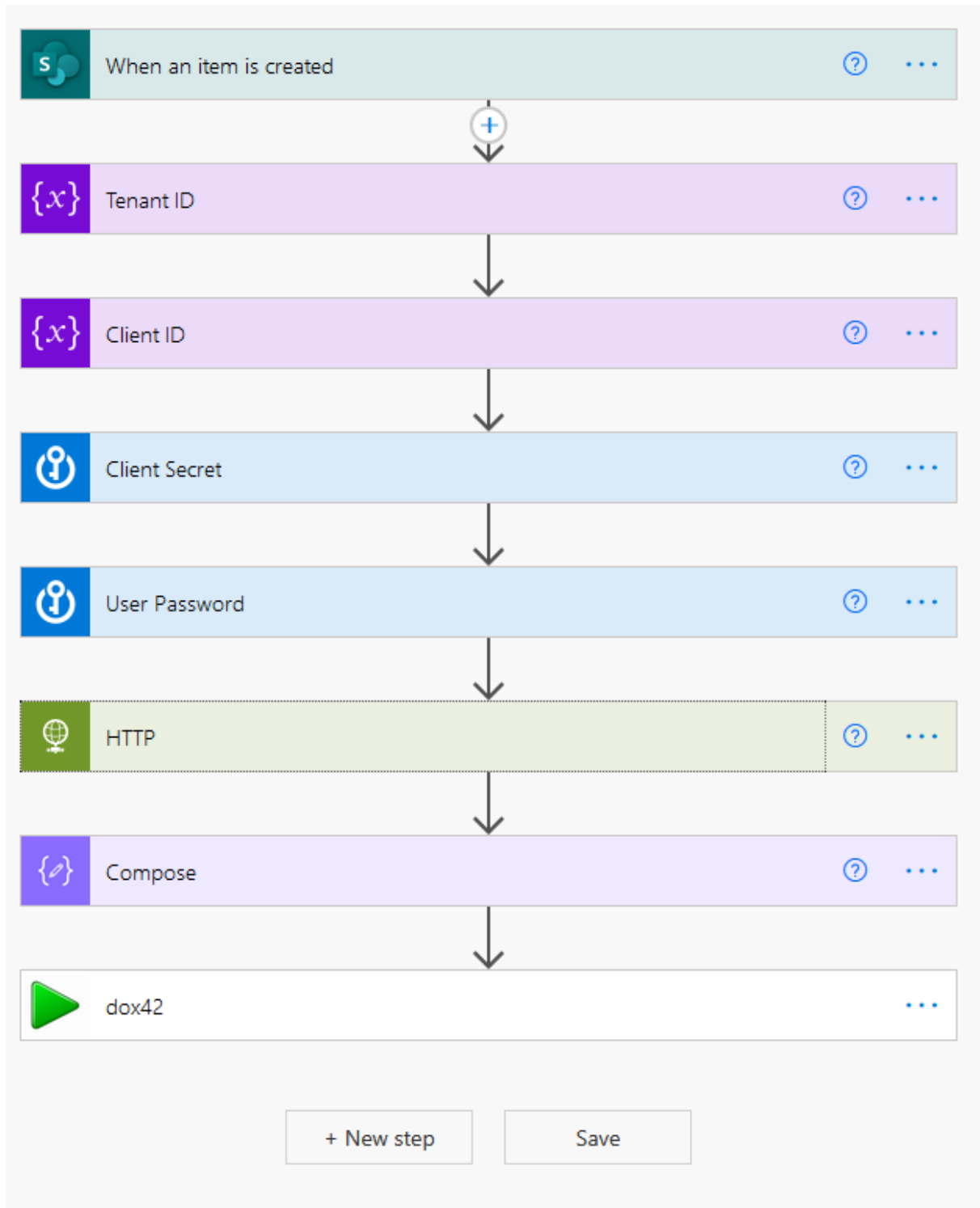


dox42 Service Call POST



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 Overview page for an application named 'dox42 Online'. The left sidebar contains navigation links: Overview (selected), Quickstart, Integration assistant, Manage (with sub-links for Branding, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, and App roles), and a minus sign. The main content area has a search bar and action buttons (Delete, Endpoints, Preview features). A warning banner states: 'A certificate or secret is expiring soon. Create a new one →'. Below this is the '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. At the bottom, there is a blue information banner: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authn'. Navigation links 'Get Started' and 'Documentation' are at the bottom.

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	

Settings

- Keys
- Secrets**
- Certificates
- Access policies
- Networking
- Security
- Properties
- Locks

Monitoring

- Alerts
- Metrics
- Diagnostic settings

4.3 Configuring the Flow

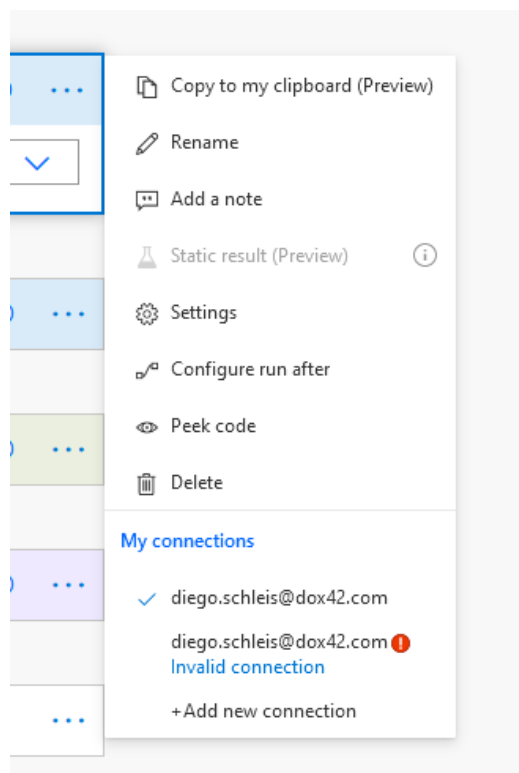
Now that we have the necessary HTTP request parameters with values, we can start to implement them in the flow. You have already seen the finished flow in the first illustration of this documentation, now we will look at each component in detail.

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.

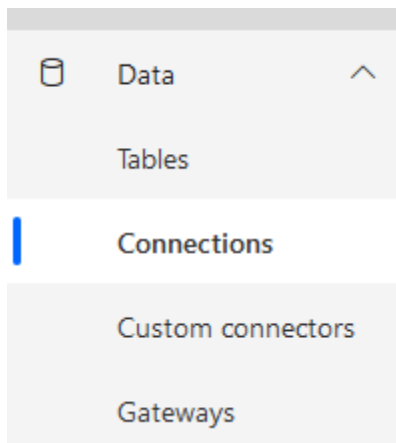
Next, we highly recommend using variables in your flow. In this example client ID and tenant ID are saved as variables, since they are not as sensitive as other credentials, but you can also save those in your Azure key vault.

The screenshot shows two variable declaration steps in a Power Automate flow. The first step is titled 'Tenant ID' and has a name 'tenantId', type 'String', and a masked value field. The second step is titled 'Client ID' and has a name 'clientId', type 'String', and a masked value field. Arrows indicate the flow from the first step to the second.

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**.



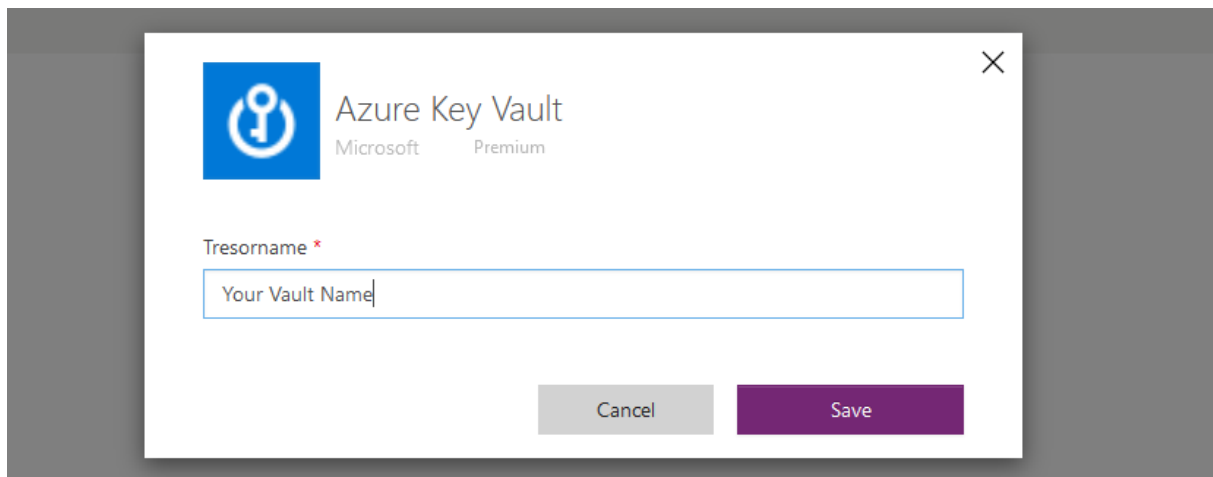
If you get an invalid connection as soon as you add the Azure key vault action, open the **Connections** list in a new browser tab. You can find said list here:



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



The screenshot shows the 'Connections' page for the user 'diego.schleis@dox42.com'. At the top, there is a toolbar with 'Edit' (pencil icon), 'Switch account' (key icon), and 'Delete' (trash icon). Below the toolbar, the breadcrumb 'Connections > diego.schleis@dox42.com' is displayed. Three tabs are visible: 'Details' (active), 'Apps using this connection', and 'Flows using this connection'. The 'Details' tab shows the following information:

- Connector name:** Azure Key Vault (with an information icon)
- Description:** Azure Key Vault ist ein Dienst zum sicheren Speichern und Zugreifen auf Geheimnisse. A green 'Premium' badge is shown below the description.
- Status:** Parameter value missing. A blue link 'Update password' is provided.
- Owner:** Diego Schleis
- Created:** 20.8.2021, 08:38:36
- Modified:** 20.8.2021, 08:38:38

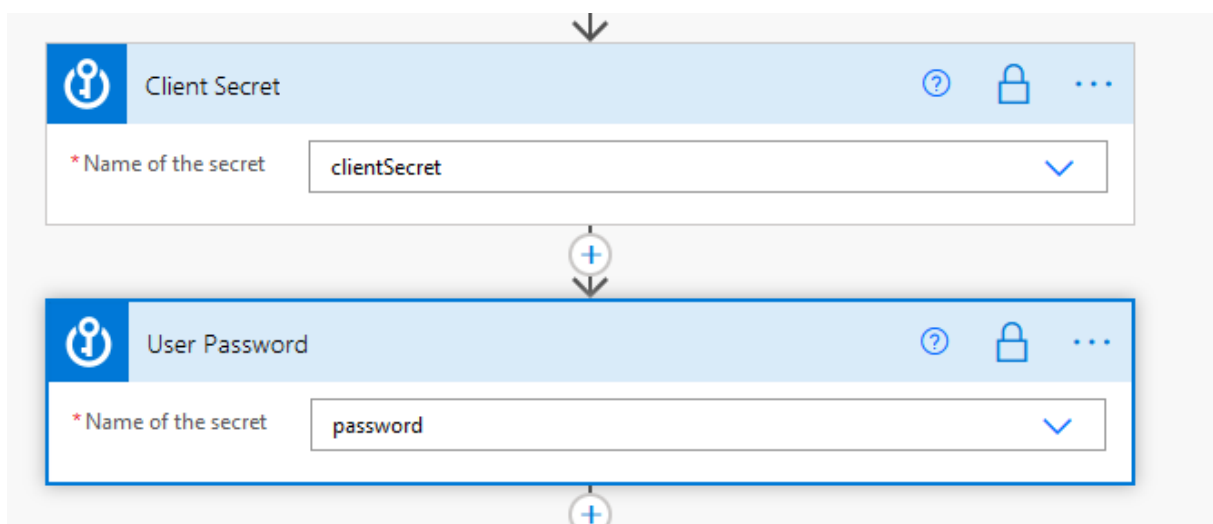


Clicking **Update password** will pop-up a Microsoft authentication window.

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:

*** Method** POST

*** URI** https://login.microsoftonline.com/{x} tenantId x /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={x} clientId x &client_secret={x} value x &scope=user.read openid profile offline_access&username=diego.schleis@dox42.com&password={x} value x

Cookie

Enter HTTP cookie

[Show advanced options](#) ▾

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']**

[Show advanced options](#) ▾

Compose

*** Inputs** fx outputs(...) x [Add dynamic content](#)

dox42

[+ New step](#) [Save](#)

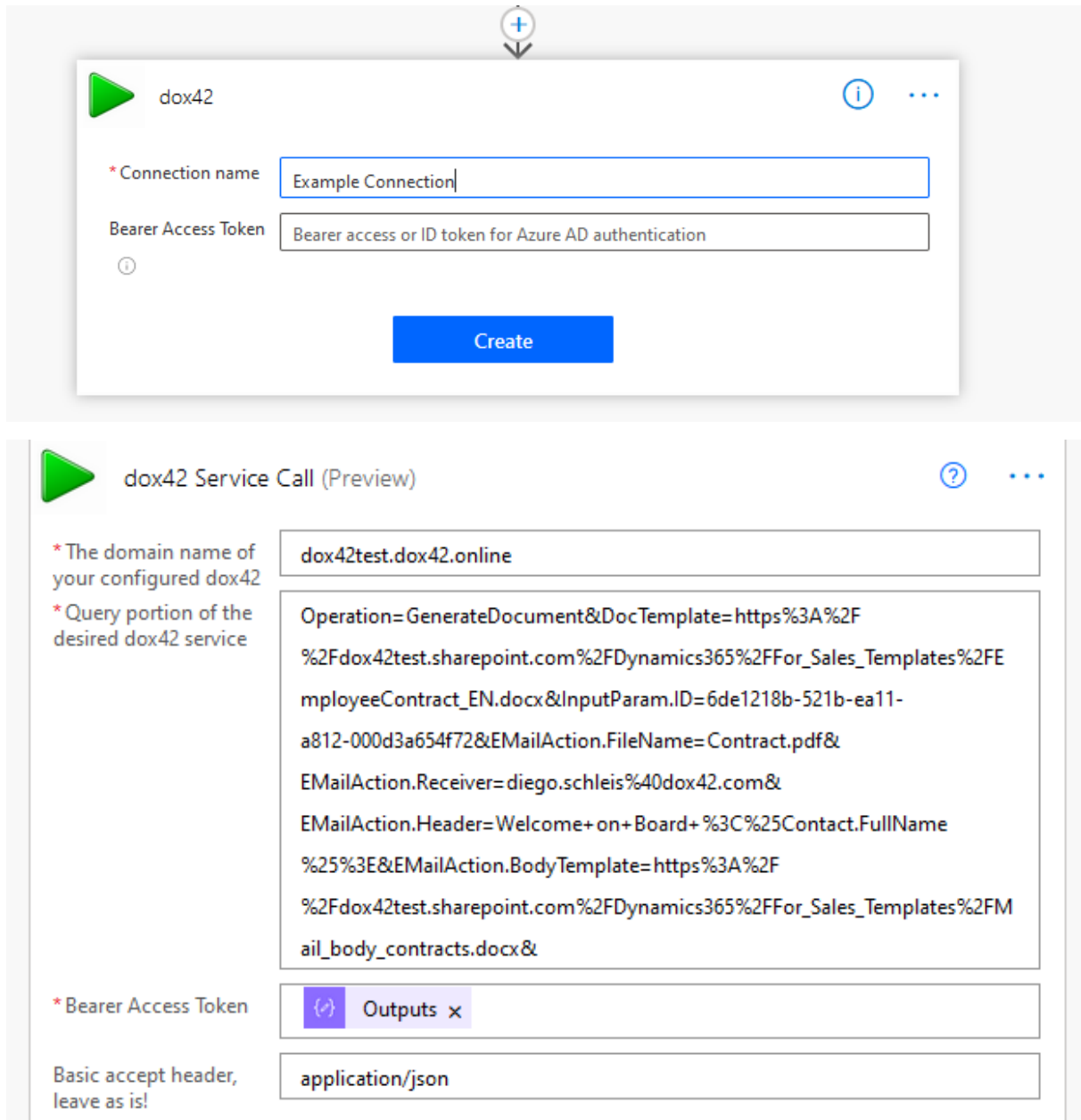
Add an expression to do basic things like access, convert, and compare values. [Learn more about dynamic content.](#)

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:



The image shows two screenshots of the dox42 connector configuration interface.

Top Screenshot: dox42 configuration panel

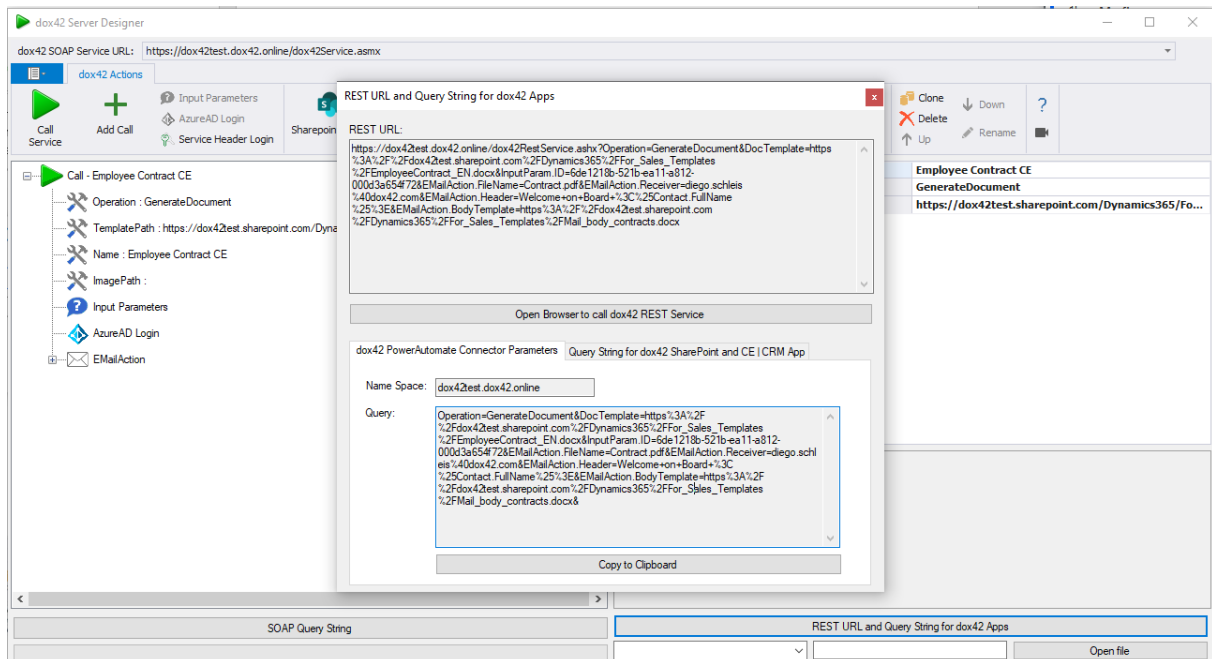
- Header:** dox42 (with a green play button icon on the left and an info icon on the right).
- Fields:**
 - * Connection name:** Example Connection
 - Bearer Access Token:** Bearer access or ID token for Azure AD authentication
- Buttons:** A blue "Create" button at the bottom.

Bottom Screenshot: dox42 Service Call (Preview) panel

- Header:** dox42 Service Call (Preview) (with a green play button icon on the left and a help icon on the right).
- Fields:**
 - * The domain name of your configured dox42:** dox42test.dox42.online
 - * Query portion of the desired dox42 service:**

```
Operation=GenerateDocument&DocTemplate=https%3A%2F%2Fdox42test.sharepoint.com%2FDynamics365%2FFor_Sales_Templates%2FEMPLOYEEContract_EN.docx&InputParam.ID=6de1218b-521b-ea11-a812-000d3a654f72&EMailAction.FileName=Contract.pdf&EMailAction.Receiver=diego.schleis%40dox42.com&EMailAction.Header=Welcome+on+Board+%3C%25Contact.FullName%25%3E&EMailAction.BodyTemplate=https%3A%2F%2Fdox42test.sharepoint.com%2FDynamics365%2FFor_Sales_Templates%2FMail_body_contracts.docx&
```
 - * Bearer Access Token:** Outputs x (with a blue icon and a close button)
 - Basic accept header, leave as is!** application/json

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

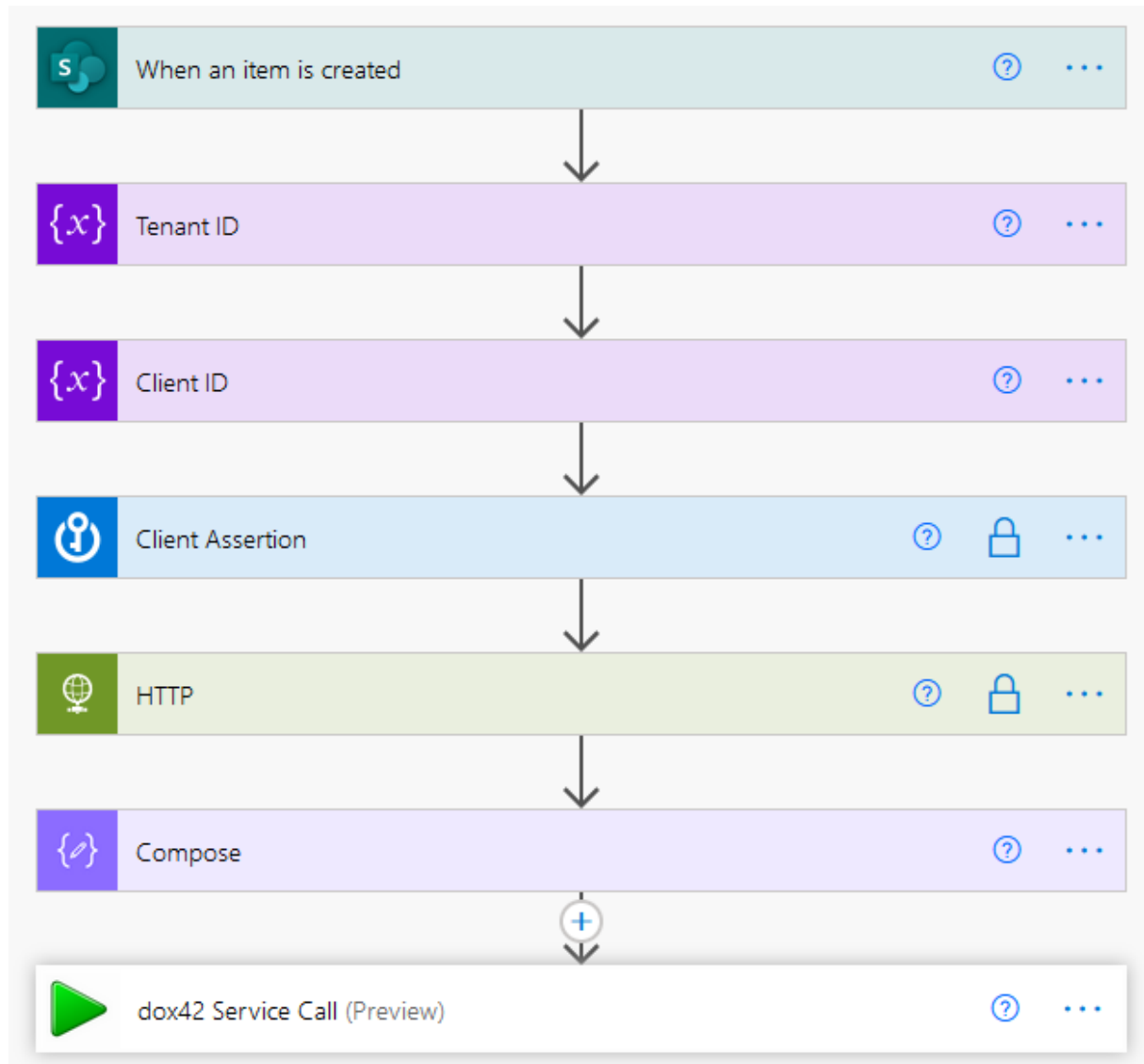


Configure a working config in the server designer and navigate to “REST URL and Query String for dox42 Apps” in the “dox42 Power Automate Connector Parameters” you will find the correct parameters for the dox42 connector. “Name Space” represents the “domainname” parameter and “Query” the “querystring”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

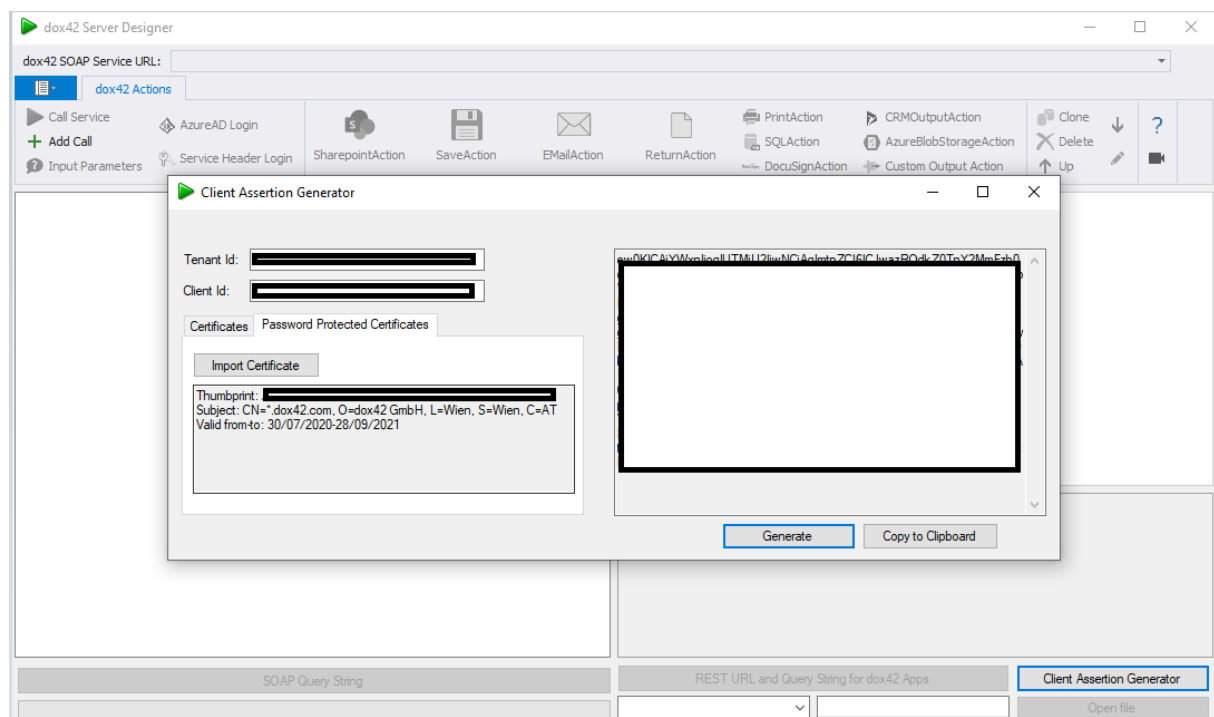
grant_type	Should be client_credentials
client_id	Your Client ID
client_assertion	A JWT token signed with a certificate
scope	For example, a SharePoint scope: yourcompany.sharepoint.com/.default
client_assertion_type	Should be urn:ietf:params:oauth:client-assertion-type:jwt-bearer

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.

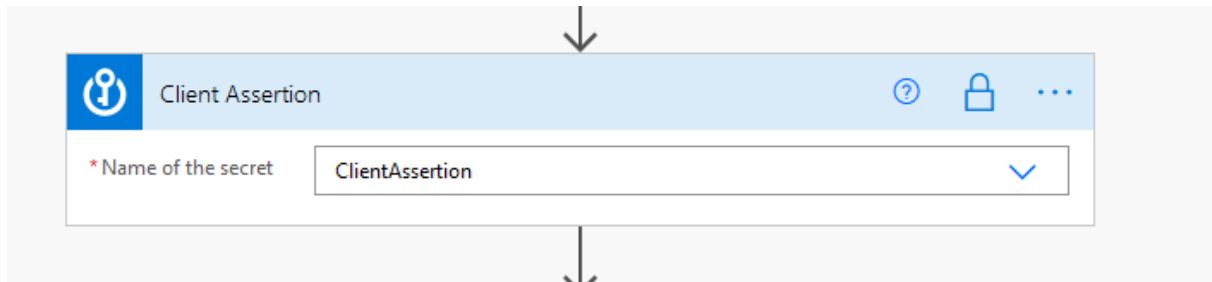


If you wish to do this step without the Server Designer you can refer to this article especially: <https://docs.microsoft.com/en-us/azure/architecture/multitenant-identity/client-assertion>

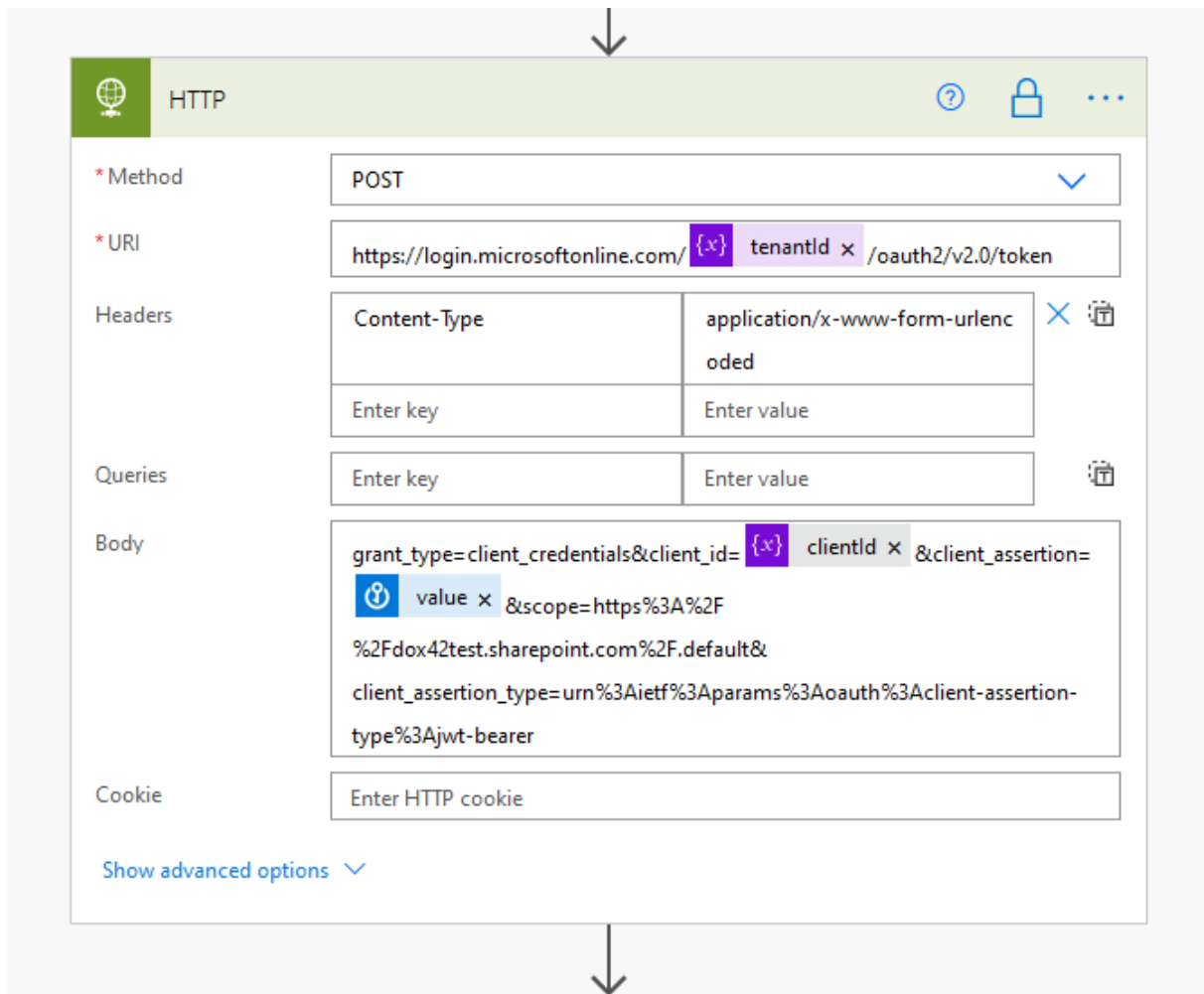
Again, we highly recommend using Azure Key Vault or similar services to encrypt secrets and passwords in your flow.

5.2 Configuring the Flow

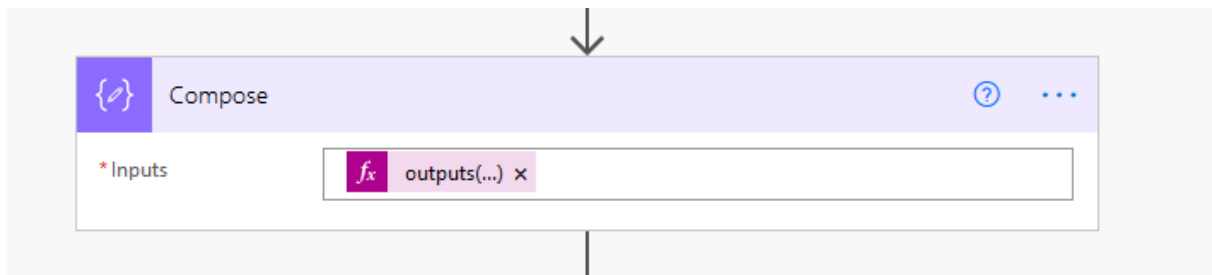
For the majority of the steps, especially Azure key vault, you can refer to method one.



Get the client assertion from Azure key vault.

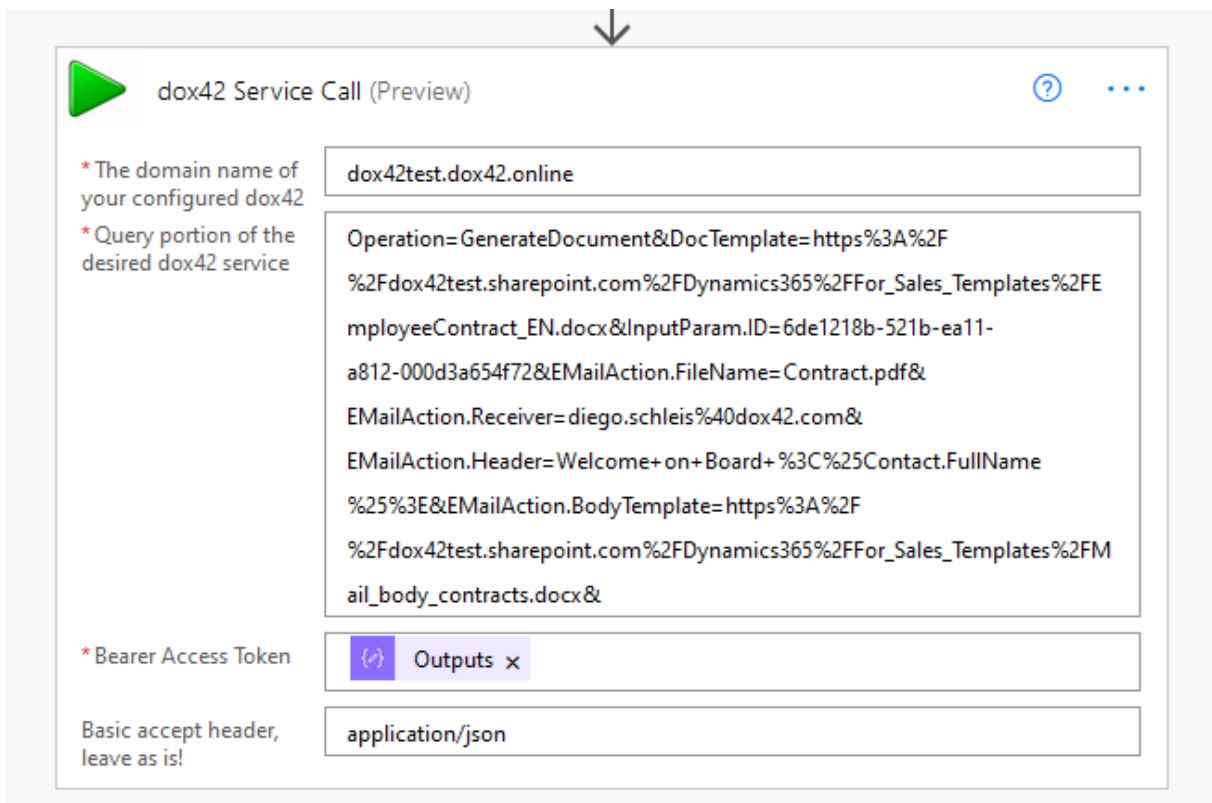


Again, the variables and secrets from the key vault are used for the HTTP request. Additionally, you have to define the **content-type** in the header with the value **application/x-www-form-urlencoded**.



To retrieve the access token from the HTTP call, enter this expression:

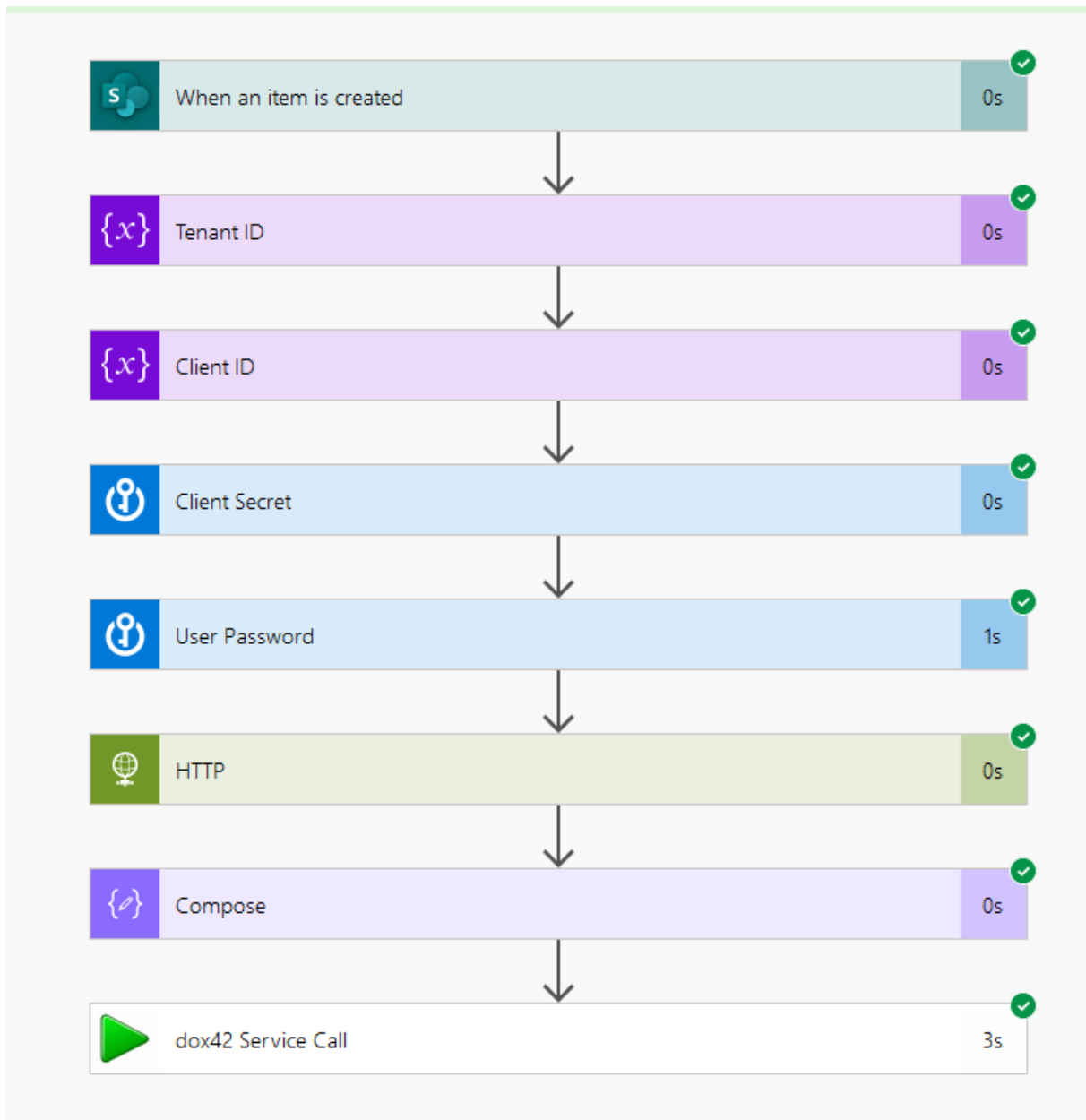
outputs('HTTP').body?['access_token']



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.