

# Prepare connector files for certification

08/12/2025

After you finish developing your Microsoft Copilot Studio and Power Platform custom connector and agent-ready connector files, you need to prepare the files for certification. As a verified or independent publisher, follow these steps to prepare the files for certification and generate the connector and agent-ready connector files you submit to Microsoft.

## ⓘ Note

This article provides information for certifying custom connectors in Azure Logic Apps, Microsoft Power Automate, Microsoft Power Apps and agent-ready connector files for Microsoft Copilot Studio. Before following the steps in this article, read [Get your connector certified](#).

## Step 1: Meet submission requirements for connectors

To maintain a high standard of quality and consistency among our certified connectors, this section outlines a set of requirements and guidelines that your custom connector must adhere to for Microsoft certification.

### Give your connector a title

The title must meet the following requirements:

- Must exist and be written in English.
- Must be unique and distinguishable from any existing connector title.
- Should be the name of your product or organization.
- Should follow existing naming patterns for certified connector. For independent publishers, the connector name should follow the pattern, `Connector Name (Independent Publisher)`.
- Can't be longer than 30 characters.
- Can't contain the words *API*, *Connector*, *Copilot Studio*, or any of our other Power Platform product names (for example, *Power Apps*).
- Can't end in a nonalphanumeric character, including carriage return, new line, or blank space.

### Examples

- Good connector titles: Azure Sentinel\*, \*Office 365 Outlook
- Poor connector titles: Azure Sentinel's Power Apps Connector, Office 365 Outlook API

## Write a description for your connector

The description must meet the following requirements:

- Ensure your description complies with the [Marketplace guidelines](#).
- Must exist and be written in English.
- Must be free of grammatical and spelling errors.
- Should describe concisely the main purpose and value offered by your connector.
- Must be longer than 30 characters and shorter than 500 characters.
- Can't contain any Copilot Studio or other Power Platform product names (for example, *Power Apps*).

## Design an icon for your connector (Only applicable for verified publishers)

This section doesn't apply to independent publishers.

- Create a logo with 1:1 dimensions within a range of 100 x 100 to 230 x 230 pixels (no rounded edges).
- Use a non-transparent, nonwhite color (#ffffff) background, and not-default color (#007ee5) that matches your specified icon background color.
- Ensure the icon is unique to any other certified connector icon.
- Submit the logo in PNG format as <icon>.png.
- Set logo dimensions below 70% for the image's height & width with a consistent background.
- Ensure brand color is a valid hexadecimal color and shouldn't be white (#ffffff) or default (#007ee5).

## Define operation and parameter summaries and descriptions

The summaries and descriptions must meet the following requirements:

- Must exist and be written in English.
- Must be free of grammatical and spelling errors.
- Operation and parameter summaries should be phrases of 80 characters or less, and contain only alphanumeric characters or parentheses.

- Operation and parameter descriptions should be full, descriptive sentences that end with punctuation.
- Must not contain any Copilot Studio or other Power Platform product names (for example, *Power Apps*).

## Define exact operation responses

The operation responses must meet the following requirements:

- Define operation responses with an exact schema only with expected responses.
- Don't use default responses with an exact schema definition.
- Provide valid response schema definitions for all operations in the swagger.
- Empty response schemas aren't allowed except in special cases where the response schema is dynamic. This means no dynamic content is shown in the output and makers must use JSON to parse the response.
- Empty operations aren't allowed.
- Remove empty properties unless they're required.

## Verify swagger properties

The properties must meet the following requirements:

- Ensure the `openapidefinition.json` is in a correctly formatted JSON file.
- Ensure swagger definition complies with the [OpenAPI Specification v2.0](#) standard and the connectors' extended standard.

## Verify connection parameters

The parameters must meet the following requirements:

- Ensure the property is updated with appropriate values for `uiDefinition` (display name, description).
- If your connection parameter uses Basic authentication, ensure JSON is formatted correctly as the following example.

JSON

```
{
  "username": {
    "type": "securestring",
    "uiDefinition": {
      "displayName": "YourUsernameLabel",
```

```

    "description": "The description of YourUsernameLabel for this
api",
    "tooltip": "Provide the YourUsernameLabel tooltip text",
    "constraints": {
      "tabIndex": 2,
      "clearText": true,
      "required": "true"
    }
  },
  "password": {
    "type": "securestring",
    "uiDefinition": {
      "displayName": "YourPasswordLabel",
      "description": "The description of YourPasswordLabel for this
api",
      "tooltip": "Provide the YourPasswordLabel tooltip text",
      "constraints": {
        "tabIndex": 3,
        "clearText": false,
        "required": "true"
      }
    }
  }
}

```

- If your connection parameter has APIKey as authentication, ensure JSON is formatted correctly as the following example.

JSON

```

{
  "api_key": {
    "type": "securestring",
    "uiDefinition": {
      "displayName": "YourApiKeyParameterLabel",
      "tooltip": "Provide your YourApiKeyParameterLabel tooltip text",
      "constraints": {
        "tabIndex": 2,
        "clearText": false,
        "required": "true"
      }
    }
  }
}

```

- If your connection parameter has Generic OAuth as authentication, ensure JSON is formatted correctly as the following example.

JSON

```
{
  "token": {
    "type": "oAuthSetting",
    "oAuthSettings": {
      "identityProvider": "oauth2",
      "scopes": [
        "scope1"
      ],
      "redirectMode": "GlobalPerConnector",
      "customParameters": {
        "AuthorizationUrl": {
          "value": "https://contoso.com"
        },
        "TokenUrl": {
          "value": "https://contoso.com"
        },
        "RefreshUrl": {
          "value": "https://contoso.com"
        }
      },
      "clientId": "YourClientId"
    },
    "uiDefinition": null
  }
}
```

- If your connection parameter has OAuth2 identity provider, ensure the identity provider is from the list of supported OAuth2 providers. Here's an example of GitHub OAuth2 identity provider:

JSON

```
{
  "token": {
    "type": "oAuthSetting",
    "oAuthSettings": {
      "identityProvider": "github",
      "scopes": [
        "scope1"
      ],
      "redirectMode": "GlobalPerConnector",
      "customParameters": {},
      "clientId": "YourClientId"
    },
    "uiDefinition": null
  }
}
```

#### ⓘ Note

If your connector is using OAuth, it's important to regularly monitor and renew expiring client ID and client secret credentials so customers are able to continue using your connector. Be sure you submit the connector update one month prior to the date your client ID and client secret are set to expire.

- If your connection parameter has Microsoft Entra ID as authentication, ensure JSON is formatted correctly as the following example.

JSON

```
{
  "token": {
    "type": "oAuthSetting",
    "oAuthSettings": {
      "identityProvider": "aad",
      "scopes": [
        "scope1"
      ],
      "redirectMode": "GlobalPerConnector",
      "customParameters": {
        "LoginUri": {
          "value": "https://login.microsoftonline.com"
        },
        "TenantId": {
          "value": "common"
        },
        "ResourceUri": {
          "value": "resourceUri"
        },
        "EnableOnbehalfOfLogin": {
          "value": false
        }
      },
      "clientId": "AzureActiveDirectoryClientId"
    },
    "uiDefinition": null
  }
}
```

## Create quality English language strings

Connectors are localized as part of Power Automate localization; therefore, when you develop a connector, the quality of the English language strings is key to the translation quality. Here are some major areas to focus on as you create the values of the strings you provide.

- Run a spell check program to ensure all string values are free of typographical errors. If there's any incomplete English language string, the translation result is incomplete or

incorrect in context.

- Make sure the sentence is in complete form—meaning it has at least a subject and a predicate. If the sentence isn't complete, that can also generate lower quality translations.
- Make sure the meaning of the sentence is clear. If the meaning of the sentence is ambiguous, that can also generate lower quality or incorrect translations.
- Make sure summaries, x-ms-summaries, and descriptions are grammatically correct. Don't copy and paste the summaries. To learn how they show within the product, go to [Connector string guidance](#).
- Avoid runtime composite strings, if possible. Use fully formed sentences instead. Concatenated strings or sentences make it difficult to translate, or can cause a wrong translation.
- Make sure to capitalize abbreviations to make them clear. Capitalization reduces the chance of it being mistaken for a typographical error.
- Fix CAMEL form strings if you want to localize the string value. Strings in CaMeL form (for example, *minimizeHighways* or *MinimizeHighways*) are typically considered nontranslatable.

## Step 2: Run the Solution Checker to validate your connector

Solution Checker is a mechanism to conduct static analysis to ensure your connector adheres to Microsoft certification standards. Add your connector to a solution in Power Automate or Power Apps and then follow the instructions in [Validate a custom connector with solution checker](#) to run the solution checker.

Watch this video to learn how to run the Solution Checker.

<https://learn-video.azurefd.net/vod/player?id=733f72a0-9d3e-479f-b653-380c3203b042&locale=en-us&embedUrl=%2Fconnectors%2Fcustom-connectors%2Fcertification-submission>

## Step 3: Meet submission requirements for connector actions

If you're also submitting the associated connector actions for certification, be sure to follow all the guidelines in these articles:

- [Extend Microsoft 365 Copilot for Copilot agents with connector actions](#) outlines the steps to author connector actions.
- [Responsible AI guidelines](#) outlines what all AI-enabled connectors need to follow.
- [100.10 Inappropriate content](#) highlights a section in the Microsoft Commercial Marketplace Terms of Use that outlines the standards by which all AI-enabled connectors must comply. Connector actions must not generate, contain, or provide access to inappropriate, harmful, or offensive AI-generated content.

## Step 4: Prepare the connector, agent-ready connector, and related artifacts

### ⓘ Note

- Follow all the specifications to ensure the quality of your connector and agent-ready connector before certification. Failure to do so results in certification delays because you are asked to make changes.
- Provide a **production version of the host URL**. Staging, dev, and test host URLs aren't allowed.

You submit a set of files to Microsoft, which is a Solution generation from the maker portal or Microsoft Copilot Studio. To package your files, follow the steps in this section.

## Connector and agent-ready packaging guide

The procedures in this section guide you through various scenarios of packaging. If you want to package only a custom connector, use the first scenario. If you want to package both a custom connector *and* agent-ready connector, use the second scenario. If you want to package an *existing* connector and agent-ready connector, use the last scenario.

- [Package your custom connector and submit for certification](#)
- [Package your custom connector and agent-ready connector for certification](#)
- [Package existing certified connector and agent-ready connector for certification](#)

### Package your custom connector and submit for certification

1. [Create a custom connector in a solution](#).
2. [Run solution checker](#) on your connector solution in step 1.



3. Export the connector solution.
4. [Create a test flow using the newly created custom connector or, add an existing flow into the solution.](#)
5. Export the flow solution.
6. [Create a package with the solutions](#) from steps 3 and 5.
7. [Create an intro.md file.](#)
8. Create the final package as a zip file in the following format:

```
ConnectorPackage.zip -
- intro.md
- package.zip -
  PkgAssets -
    - FlowSolution.zip -
      - Connector
      - Workflows
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
    - ConnectorSolution.zip -
      - Connector
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
```

#### ⓘ Note

The names of the folder and files outside the solution are only for reference—you can choose as per your requirements. However, don't manipulate the files inside the solution.

1. Upload the package to a storage blob and [generate SAS URL](#). Ensure that your SAS URI is valid for at least 15 days.
2. Submit your package to [Partner Center](#) .

## Package your custom connector and agent-ready connector for certification

1. Follow steps 1 through 5 in [Package your custom connector and submit for certification](#) in this article.
2. [Create a plugin in the Microsoft Copilot Studio portal and export it as a solution.](#)
3. [Create a package](#) from the following:

- [Run solution checker](#) (step 2 in [Package your custom connector and submit for certification](#)).
- Export the flow solution (step 5 in [Package your custom connector and submit for certification](#)).
- [Create a plugin in Microsoft Copilot Studio and export it as a solution](#) (step 2 in this procedure).

4. [Create an intro.md file](#).

5. Create the final package as a zip file, which is in the following format.

```
ConnectorPackage.zip -
- intro.md
- package.zip -
  PkgAssets -
    - FlowSolution.zip -
      - Connector
      - Workflows
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
    - ConnectorSolution.zip -
      - Connector
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
    - AIPluginSolution.zip -
      - Connector
      - aipluginoperations
      - aiplugins
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
```

#### ⓘ Note

The names of the folder and files outside the solution are only for reference—you can choose as per your requirements. However, don't manipulate the files inside the solution.

1. Upload the package to a storage blob and [generate SAS URL](#). Ensure that your SAS URI is valid for at least 15 days.
2. Submit your package to [Partner Center](#) .

## Package existing certified connector and agent-ready connector for certification

1. Create a solution in [Power Automate](#) and add the already certified connector to it.

2. Follow steps 2 through 4 in [Package your custom connector and submit for certification](#) in this article.
3. [Extend Microsoft 365 Copilot or Copilot agents with connector actions](#).
4. Export the plugin as solution.
5. [Create a package](#) from the following:
  - [Run solution checker](#) on your connector solution (step 2 in [Package your custom connector and submit for certification](#) in this article).
  - [Create a plugin in Copilot Studio and export it as a solution](#) (step 3 in this procedure).
  - [Create a plugin in Copilot Studio and export it as a solution](#) (step 4 in this procedure).
6. [Create an intro.md file](#).
7. Create the final package as a zip file, which is in the following format.

```
ConnectorPackage.zip -
- intro.md
- package.zip -
  PkgAssets -
    - FlowSolution.zip -
      - Connector
      - Workflows
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
    - ConnectorSolution.zip -
      - Connector
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
    - AIPluginSolution.zip -
      - Connector
      - aipluginoperations
      - aiplugins
      - customizations.xml
      - [Content_Types].xml
      - solution.xml
```

### ⓘ Note

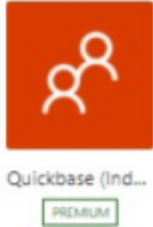
The names of the folder and files outside the solution are only for reference—you can choose as per your requirements. However, don't manipulate the files inside the solution.

1. Upload the package to a storage blob and [generate SAS URL](#). Ensure that your SAS URI is valid for at least 15 days.

## 2. Submit your package to [Partner Center](#) .

Both verified publishers and independent publishers download `openapidefinition.json` in their artifacts. You need to set the `IconBrandColor` in this file.

- **Verified publishers:** Set `iconBrandColor` to your brand color in the `openapidefinition` file.
- **Independent publishers:** Set `iconBrandColor` to `"#da3b01"` in the `openapidefinition` file.



## Create an `intro.md` artifact

An `intro.md` file is necessary for both independent publishers and verified publishers. You need to create an `intro.md` file to document your connector's features and functionality. For an example of documentation to include, go to the [Readme.md example](#) . To learn about writing an `intro.md` file, look at other `intro.md` files (also known as `Readme.md` files) in our [GitHub repository](#) .

If you're an independent publisher and your connector uses OAuth, make sure you include instructions for how to obtain credentials.

### Tip

**Known Issues and Limitations** is a great section to maintain to keep your users up-to-date.

## Step 5: Validate the package for structure

The package validation script validates the package structure and helps you generate the package in an acceptable format for certification. Download the package validator script, [ConnectorPackageValidator.ps1](#) .

### Important

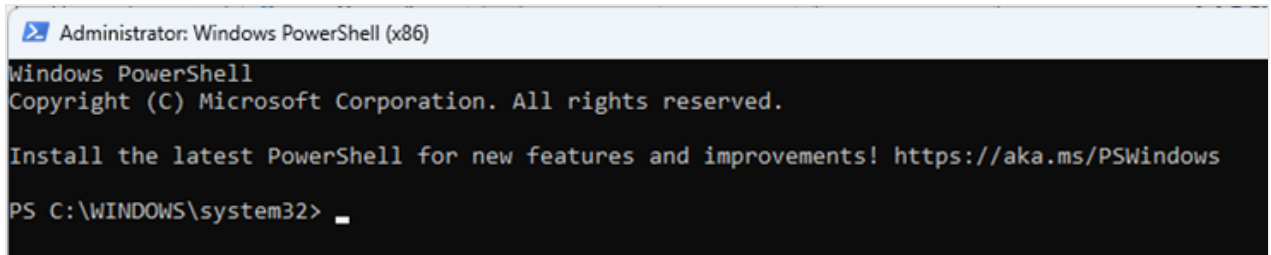
If you use **macOS**, you need to [Install PowerShell on macOS](#).

If you're *not* a Microsoft 365 or Windows user, take the steps in [What is Microsoft Entra ID](#) to create an **Entra ID** to generate a SAS URL for your package and authenticate into

Partner Center to get certification notices.

To run the package validation script, follow these steps:

1. Open Windows PowerShell in Admin mode.



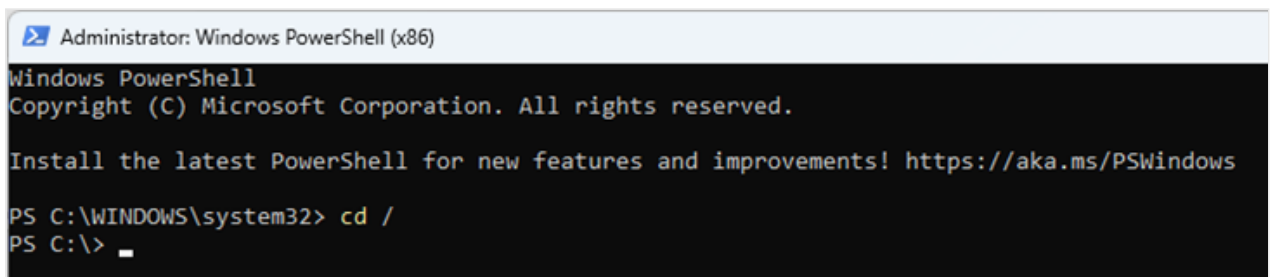
```
Administrator: Windows PowerShell (x86)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> _
```

2. Change the drive location by entering `cd /`.

The following example uses `C:\`.



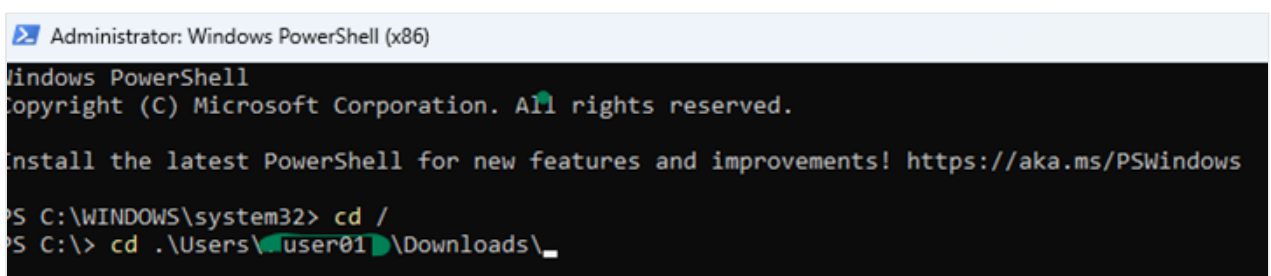
```
Administrator: Windows PowerShell (x86)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd /
PS C:\> _
```

3. Go to the path where you downloaded the package validator script.

For example, if the path is `C:\Users\user01\Downloads`, you enter `cd .\Users\user01\Downloads\`.



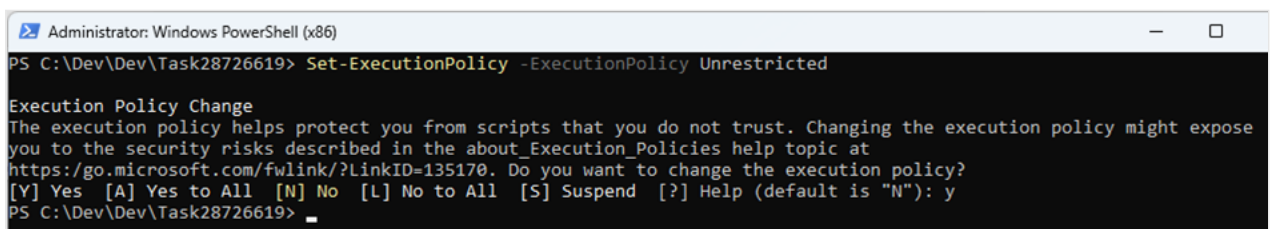
```
Administrator: Windows PowerShell (x86)
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd /
PS C:\> cd .\Users\user01\Downloads\
PS C:\Users\user01\Downloads> _
```

4. Set the execution policy to unrestricted by entering the following command:

`Set-ExecutionPolicy -ExecutionPolicy Unrestricted`



```
Administrator: Windows PowerShell (x86)
PS C:\Dev\Dev\Task28726619> Set-ExecutionPolicy -ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about_Execution_Policies help topic at https://go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): y
PS C:\Dev\Dev\Task28726619> _
```

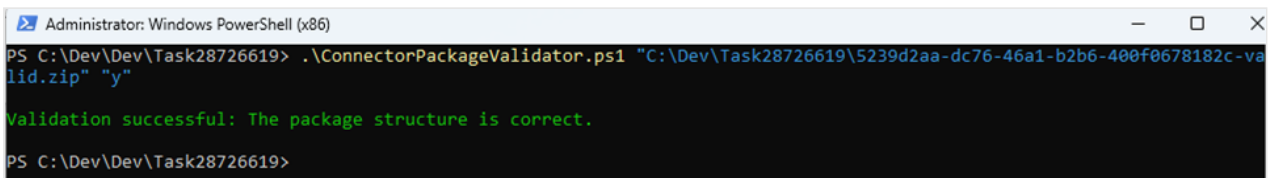
This command enables the PowerShell to be executed without any restriction.

5. Confirm your entry by entering **Y**, which stands for *Yes*.
6. Execute the **ConnectorPackageValidator.ps1** with the following steps:
  - a. Enter the zip file path that contains the connector package.
  - b. Specify whether or not the AI action is enabled.

As shown in the following example, the first argument is a valid zip file path that contains the package. The second argument is `yes / y` to indicate the AI action is enabled, or `no / n` to indicate that it's disabled.

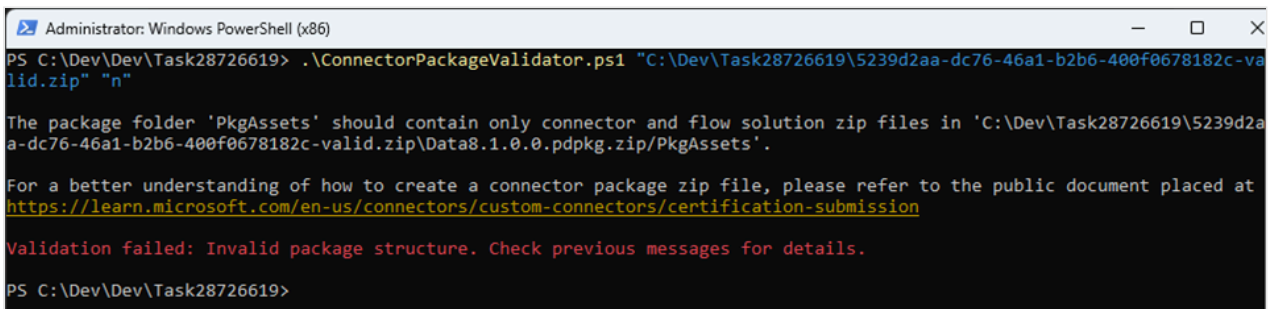
```
.\ConnectorPackageValidator.ps1  
"C:\Users\user01\Downloads\ConnectorCertificationPackage.zip" "no"
```

If the package structure is correct, the following success message displays:



```
Administrator: Windows PowerShell (x86)  
PS C:\Dev\Dev\Task28726619> .\ConnectorPackageValidator.ps1 "C:\Dev\Task28726619\5239d2aa-dc76-46a1-b2b6-400f0678182c-valid.zip" "y"  
Validation successful: The package structure is correct.  
PS C:\Dev\Dev\Task28726619>
```

If there's an issue with the package structure, the script provides issue details by detecting and highlighting the defects in the package structure.



```
Administrator: Windows PowerShell (x86)  
PS C:\Dev\Dev\Task28726619> .\ConnectorPackageValidator.ps1 "C:\Dev\Task28726619\5239d2aa-dc76-46a1-b2b6-400f0678182c-valid.zip" "n"  
  
The package folder 'PkgAssets' should contain only connector and flow solution zip files in 'C:\Dev\Task28726619\5239d2aa-dc76-46a1-b2b6-400f0678182c-valid.zip\Data8.1.0.0.pdpkg.zip\PkgAssets'.  
  
For a better understanding of how to create a connector package zip file, please refer to the public document placed at https://learn.microsoft.com/en-us/connectors/custom-connectors/certification-submission  
  
Validation failed: Invalid package structure. Check previous messages for details.  
PS C:\Dev\Dev\Task28726619>
```

## Step 6: Submit your connector and/or AI-enabled connector for certification

During the submission process, you open-source your connector and/or AI-enabled connector to our [Microsoft Power Platform Connectors repository](#).

1. (For independent publishers) To submit the package to Microsoft for certification, follow the instructions in [Independent Publisher Certification process](#).
2. (For verified publishers) To submit the package to Microsoft for certification into Partner Center, follow the instructions in [Verified Publisher Certification Process](#).

If you're a verified publisher, you need to submit a script.csx file if you're using custom code.

If your connector has OAuth, provide Client ID and Secret in [Partner Center](#). Also, get the APIName from your connector submission request to update your app.

As part of submission, Microsoft certifies and/or plugin your connector. If you need to troubleshoot swagger errors, go to [Fix Swagger Validator errors](#).

## Checklist before submitting

Before moving on to [Submit your connector for Microsoft certification](#), ensure that:

- Your connector and agent-ready connector satisfy all standards set in [Step 1: Meet submission requirements for connectors](#) and [Step 3: Meet submission requirements for connector actions](#).
- No operations are missing [summary](#), [description](#), or [visibility information](#).
- You tested your custom connector and agent-ready connector to ensure the operations work as expected (at least 10 successful calls per operation).
- No runtime or schema validation errors appear in the [test section](#) of the custom connector wizard.

## For queries regarding certification

You need to have Microsoft Teams to join the Office Hours meeting. If you need access, view your options in [Microsoft Teams](#).

Join [Office Hours Meeting](#) every Tuesday at 3:30 PM to 4:30 PM, UTC (Coordinated Universal Time).

### Tip

- Create YouTube videos, blogs, or other content to share samples or screenshots for how to get started with the connector and agent-ready connector.
- Include the links in the [intro.md](#) file so that we can add to our docs.
- Add [tooltips](#) to your swagger file to help your users be more successful.

## Next step

- [Verified publisher certification process](#)
- [Independent publisher certification process](#)