Creating a PDF Connector - Developer’s Guide

Contents

[Overview 3](#_Toc203582522)

[Key Files and Their Purpose 3](#_Toc203582523)

[✅ Menu.cs 3](#_Toc203582524)

[✅ Connector.cs 3](#_Toc203582525)

[✅ PropertySheet2.cs 3](#_Toc203582526)

[✅ Resources.resx 3](#_Toc203582527)

[Adding a Menu in PPDF 4](#_Toc203582528)

[File: Menu.cs 4](#_Toc203582529)

[Steps to Add a Custom Menu Item: 4](#_Toc203582530)

[Required Methods to Implement in Connector.cs 5](#_Toc203582531)

[Changes in PropertySheet2.cs 6](#_Toc203582532)

[Change GuidAttribute 6](#_Toc203582533)

[Build Settings 7](#_Toc203582534)

[Release 7](#_Toc203582535)

[Deploying the Connector 8](#_Toc203582536)

[Copy DLL and its Dependencies 8](#_Toc203582537)

[Modify PublishMode.xml 8](#_Toc203582538)

[Register DLL 9](#_Toc203582539)

[Clear Cache 9](#_Toc203582540)

## Overview

This document outlines the essential components and steps involved in creating a custom PDF connector using the **CurrentDocumentBluePrint** template. This general guide helps developers understand the extensibility points and where to plug in their logic.

The CurrentDocumentBluePrint is a .NET Framework-based project that integrates into Power PDF to extend functionality for custom actions such as opening, saving, and processing PDF documents.

## Key Files and Their Purpose

### ✅ Menu.cs

* Defines custom toolbar/menu items via MenuItemDefinition.
* Handles icon resources and user interaction.

### ✅ Connector.cs

* **Implements** DMSConnecor interface.
* Entry point for document events: DocAddNew, DocOpen, DocSave.
* Handles initialization and global logic.

### ✅ PropertySheet2.cs

* Optional configuration dialog.
* Can be used to collect or store user preferences like settings, URLs.

### ✅ Resources.resx

* Stores icons and string resources used in the UI.

## Adding a Menu in PPDF

### File: Menu.cs

This file contains the MenuItemDefinition class and a static array menuDefinitions that defines what menu items will appear in the **Power PDF** toolbar or menu for your connector.

### Steps to Add a Custom Menu Item:

1. **Define an entry in enum:**

public enum ItemId  
{  
 OPEN = 1,  
 SAVE = 2,  
 CUSTOMACTION = 3  
}

1. **Add a Resource String:** In Resources.resx (under Resources folder), add:
   * MenuCustomAction → value: "Run Custom Task"
2. **Add Bitmap Resources if required:**
   * Add icons (e.g., Image\_Custom.png) to your Resources.resx.
3. **Add Entry to menuDefinitions :**

new MenuItemDefinition(ItemId.CUSTOMACTION, "MenuCustomAction",  
 "Perform a custom task", true, CallbackType.CALLBACK\_MENUITEM,  
 "Image\_Custom", "Image\_Custom\_150", "Image\_Custom\_200",  
 "Image\_Custom\_Small", "Image\_Custom\_Small\_150", Image\_Custom\_Small\_200", true)

1. **Make sure your image names match** what’s used in Resources.resx. If required add new icons to Resources.resx.

## Required Methods to Implement in Connector.cs

Following methods can be implemented or added code as required :

* string **DocAddNew**(string sourceFile, string title, string[] docProperties)
* This is the first method to be executed if added Menu Item is of CallBackType -> CallBack\_Save
* Here you have access to document path and other document properties.
* All logic related to document processing can be placed here.
* Void **DocOpen**(string docId, OpenMode mode)
* Gets triggered when Callbacktype of CallbackType.CALLBACK\_OPEN is used.
* It's your connector's way of **handling "open" behavior**, such as importing, modifying, or analyzing a PDF before Power PDF displays it.
* void IDMSConnector.**DocSave**(string docId, string targetFileName, out string newDocId)
* Tied to menu items defined with CallbackType.CALLBACK\_SAVE or CallbackType.CALLBACK\_SAVEAS.
* This method gives your connector control over **how** and **where** the document is saved when the user chooses **Save** or **Save As** through your connector.
* void IDMSConnector.**MenuAction**(int menuItemId, string docId)string GetConnectorVersion()
* When the user clicks a custom **menu item or toolbar button** that you’ve added to Power PDF’s UI via Menu.cs.
* The value of menuItemId will correspond to the ItemId enum values you’ve defined

## Changes in PropertySheet2.cs

* The PropertySheet2.cs file is responsible for defining the configuration UI that appears when the user selects: **File → Options → [Your Connector Name]**
* Here you can define all UI controls you would like to see as part of settings to store values like API URL, Password etc.
* CurrentDocumentBluePrint solution has two fields API URL and Secret storing values to Registry.

## Change GuidAttribute

In Connector.cs change GUIDAttribute to a new GUID by generating one using Tools -> Create GUID

A screenshot of a computer

AI-generated content may be incorrect.

## Build Settings

To ensure compatibility:

* Set Platform Target to x86 under: **Project → Properties → Build → Platform target**
* Optionally, set Assembly Version to AssemblyVersion("1.0.\*")] and Sign the assembly by providing strong name file.

## Release

Make sure to select Solution Configurations is set to ‘Release’ and build to generate YourConnector.dll in Release folder.

# Deploying the Connector

## Copy DLL and its Dependencies

Copy the following to the Power PDF installation directory: C:\Program Files (x86)\Kofax\Power PDF\bin\Connectors

* YourConnector.dll
* Any dependencies (e.g., System.Text.Json.dll if needed)

## Modify PublishMode.xml

* **Power PDF reads this file when loading connectors.**
* It determines **which UI actions are available** to the user.
* All the entries added in Menu.cs mode should be added to this file itemID starting from 0

Eg:

Menu.cs

new MenuItemDefinition( ItemId.TryMe, "MenuTryMe", "", true, CallbackType.CALLBACK\_MENUITEM,"Image\_Open", "Image\_Open\_150", "Image\_Open\_200", "Image\_Open\_Small", "Image\_Open\_Small\_150", "Image\_Open\_Small\_200", true)

## 

Corresponding change in Publish Mode.xml

These lines are added in the section <toolbar name="connectors" shortKey ="N">

*<PFFGroup name="connector::CurrentDocumentBluePrint" GroupType="PFFTitleBlock" autoZip="0">*

*<PFFButton name="connector::tool::CurrentDocumentBluePrint:menuItem0"/>*

*</PFFGroup>*

## Register DLL

* Open Command Prompt with Administrator
* Run the following command

"C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe" "C:\Program Files (x86)\Kofax\Power PDF 51\bin\Connectors\YourConnector.dll" /codebase

If you want to unregister for any reason use the below command

"C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe" /unregister "C:\Program Files (x86)\Kofax\Power PDF 51\bin\Connectors\YourConnector.dll"

**Troubleshooting step:**

Registering the DLL will create a key in registry here.

HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\ScanSoft\Connectors\YourConnector.Connector

## Clear Cache

Delete Publish.xml if available at this folder. %appdata%\Kofax\PDF\PowerPDF\UILayout\Publish.xml

Make sure to Restart Power PDF before testing the changes.