Office versions and requirement sets

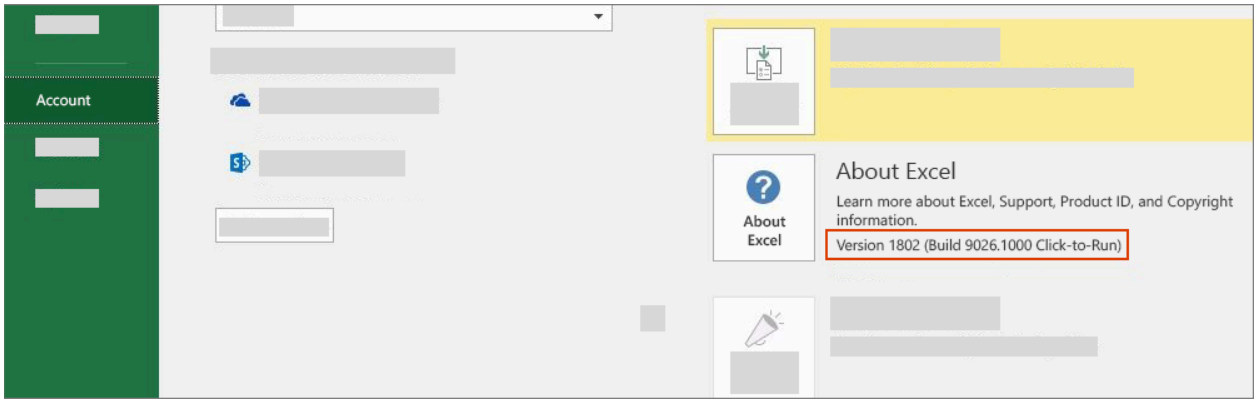
Article • 03/12/2025

There are many versions of Office on several platforms, and they don't all support every API in Office JavaScript API (Office.js). Office 2013 on Windows was the earliest version of Office that supported Office Add-ins. You may not always have control over the version of Office your users have installed. To handle this situation, we provide a system called requirement sets to help you determine whether an Office application supports the capabilities you need in your Office Add-in.

> **Note**
>
> - Office runs across multiple platforms, including Windows, in a browser, Mac, and iPad.
> - Examples of Office applications are Office products: Excel, Word, PowerPoint, Outlook, OneNote, and so forth.
> - Office is available by a Microsoft 365 subscription or perpetual license. The perpetual version is available by volume-licensing agreement or retail.
> - A requirement set is a named group of API members, for example, `ExcelApi 1.5`, `WordApi 1.3`, and so on.

How to check your Office version

To identify the Office version that you're using, from within an Office application, select the **File** menu, and then choose **Account**. The version of Office appears in the **Product Information** section. For example, the following screenshot indicates Office Version 1802 (Build 9026.1000).



> **Note**

If your version of Office is different from this, see What version of Outlook do I have? or About Office: What version of Office am I using? to understand how to get this information for your version.

Deployment

How your add-in is deployed can affect your add-in's availability on the various platforms and clients. To learn more about deployment options, see Deploy and publish Office Add-ins.

Office requirement sets availability

Office Add-ins can use API requirement sets to determine whether the Office application supports the API members that it need to use. Requirement set support varies by Office application and the Office application version (see earlier section How to check your Office version).

Some Office applications have their own API requirement sets. For example, the first requirement set for the Excel API was `ExcelApi 1.1` and the first requirement set for the Word API was `WordApi 1.1`. Since then, multiple new ExcelApi requirement sets and WordApi requirement sets have been added to provide additional API functionality.

In addition, other functionality such as add-in commands (ribbon extensibility) and the ability to launch dialog boxes (Dialog API) were added to the Common API. Add-in commands and Dialog API requirement sets are examples of API sets that various Office applications share in common.

An add-in can only use APIs in requirement sets that are supported by the version of Office application where the add-in is running. To know exactly which requirement sets are available for a specific Office application version, refer to the following application-specific requirement set articles.

- Excel JavaScript API requirement sets (ExcelApi)
- OneNote JavaScript API requirement sets (OneNoteApi)
- Outlook JavaScript API requirement sets (Mailbox)
- PowerPoint JavaScript API requirement sets (PowerPointApi)
- Word JavaScript API requirement sets (WordApi)

Some requirement sets contain APIs that can be used by several Office applications. For information about these requirement sets, see Office Common API requirement sets.

The version number of a requirement set, such as the "1.1" in `ExcelApi 1.1`, is relative to the Office application. The version number of a given requirement set (e.g., `ExcelApi 1.1`) does not correspond to the version number of Office.js or to requirement sets for other Office applications (e.g., Word, Outlook, etc.). Requirement sets for the different Office applications are released at different rates. For example, `ExcelApi 1.5` was released before the `WordApi 1.3` requirement set.

The Office JavaScript API library (Office.js) includes all requirement sets that are currently available. While there is such a thing as requirement sets `ExcelApi 1.3` and `WordApi 1.3`, there is no `Office.js 1.3` requirement set. The latest release of Office.js is maintained as a single Office endpoint delivered via the content delivery network (CDN). For more details around the Office.js CDN, including how versioning and backward compatibility is handled, see Understanding the Office JavaScript API.

Specify Office applications and requirement sets

There are various ways to specify which Office applications and requirement sets are required by an add-in. For detailed information, see Specify Office applications and API requirements

See also

- Specify Office applications and API requirements
- Install the latest version of Office
- Overview of update channels for Microsoft 365 Apps
- Reimagine productivity with Microsoft 365 and Microsoft Teams

Requirements for running Office Add-ins

08/13/2025

This article describes the software and device requirements for running Office Add-ins.

> **Note**
>
> If you plan to publish your add-in to AppSource and make it available within the Office experience, make sure that you conform to the Commercial marketplace certification policies. For example, to pass validation, your add-in must work across all platforms that support the methods that you define (for more information, see section 1120.3 and the Office Add-in application and availability page).

For a high-level view of where Office Add-ins are currently supported, see Office client application and platform availability for Office Add-ins.

Server requirements

To be able to install and run any Office Add-in, you first need to deploy the manifest and webpage files for the UI and code of your add-in to the appropriate server locations.

For all types of add-ins (content, Outlook, and task pane add-ins and add-in commands), you need to deploy your add-in's webpage files to a web server, or web hosting service, such as Microsoft Azure.

While not strictly required in all add-in scenarios, using an HTTPS endpoint for your add-in is strongly recommended. Add-ins that are not SSL-secured (HTTPS) generate unsecure content warnings and errors during use. If you plan to run your add-in in Office on the web or publish your add-in to AppSource, it must be SSL-secured. If your add-in accesses external data and services, it should be SSL-secured to protect data in transit. Self-signed certificates can be used for development and testing, so long as the certificate is trusted on the local machine.

> **Tip**
>
> When you develop and debug an add-in in Visual Studio, Visual Studio deploys and runs your add-in's webpage files locally with IIS Express, and doesn't require an additional web server.

For content and task pane add-ins, in the supported Office client applications - Excel, PowerPoint, Project, or Word - you also need either an app catalog on SharePoint to upload

the add-in's XML-formatted add-in only manifest file, or you need to deploy the add-in using the integrated apps portal.

To test and run an Outlook add-in, the user's Outlook email account must reside on Exchange 2016 or later, which is available through Microsoft 365, Exchange Online, or through an on-premises installation. The user or administrator installs manifest files for Outlook add-ins on that server. For Exchange on-premises installations, the following requirements apply.

- The server must be Exchange 2016 or later.
- Exchange Web Services (EWS) must be enabled and must be exposed to the Internet. Many add-ins require EWS to function properly.
- The server must have a valid authentication certificate in order for the server to issue valid identity tokens. New installations of Exchange Server include a default authentication certificate. For more information, see Digital certificates and encryption in Exchange 2016 and Set-AuthConfig.
- To access add-ins from AppSource, the client access servers must be able to communicate with AppSource.

> **Note**
>
> POP3 and IMAP email accounts in Outlook don't support Office Add-ins.

Client requirements: Windows desktop and tablet

The following software is required for developing an Office Add-in for the supported Office desktop clients or web clients that run on Windows-based desktop, laptop, or tablet devices.

- For Windows x86 and x64 desktops, and tablets such as Surface Pro:
  - The 32- or 64-bit version of Office 2016 or a later version, running on Windows 7 or a later version.
  - Excel 2016, Outlook 2016, PowerPoint 2016, Project Professional 2016, Project 2016, Word 2016, or a later version of the Office client, if you're testing or running an Office Add-in specifically for one of these Office desktop clients. Office desktop clients can be installed on premises or via Click-to-Run on the client computer.

  If you have a valid Microsoft 365 subscription and you don't have access to the Office client, you can download and install the latest version of Office.

- Microsoft Edge must be installed, but doesn't have to be the default browser. To support Office Add-ins, the Office client that acts as host uses webview components that are part of Microsoft Edge.

> **Note**
>
> - Strictly speaking, it's possible to develop add-ins on a machine that has Internet Explorer 11 (IE11) installed, but not Microsoft Edge. However, IE11 is used to run add-ins only on certain older combinations of Windows and Office versions. See Browsers and webview controls used by Office Add-ins for more details. We don't recommend using such old environments as your primary add-in development environment. However, if you're likely to have customers of your add-in that are working in these older combinations, we recommend that you support the Trident webview that's provided by Internet Explorer. For more information, see Support older Microsoft webviews and Office versions.
> - Internet Explorer's Enhanced Security Configuration (ESC) must be turned off for Office Web Add-ins to work. If you are using a Windows Server computer as your client when developing add-ins, note that ESC is turned on by default in Windows Server.

- One of the following as the default browser: Internet Explorer 11, or the latest version of Microsoft Edge, Chrome, Firefox, or Safari (Mac OS).
- An HTML and JavaScript editor such as Visual Studio Code, Visual Studio and the Microsoft Developer Tools, or non-Microsoft web development tool.

Client requirements: OS X desktop

Outlook on Mac, which is distributed as part of Microsoft 365, supports Outlook add-ins. Running Outlook add-ins in Outlook on Mac has the same requirements as Outlook on Mac itself: the operating system must be at least OS X v10.10 "Yosemite". Because Outlook on Mac uses WebKit as a layout engine to render the add-in pages, there is no additional browser dependency.

The following are the minimum client versions of Office on Mac that support Office Add-ins.

- Word version 15.18 (160109)
- Excel version 15.19 (160206)
- PowerPoint version 15.24 (160614)

Client requirements: Browser support for Office web clients and SharePoint

Any browser, except Internet Explorer, that supports ECMAScript 5.1, HTML5, and CSS3, such as Microsoft Edge, Chrome, Firefox, or Safari (Mac OS).

Client requirements: Non-Windows smartphone and tablet

Specifically for Outlook running on smartphones and non-Windows tablet devices, the following software is required for testing and running Outlook add-ins.

Expand table

Office application	Device	Operating system	Exchange account	Mobile browser
Outlook on the web (modern)[1]	• iPad 2 or later • Android tablets	• iOS 5 or later • Android 4.4 KitKat or later	On Microsoft 365, Exchange Online	• Microsoft Edge • Chrome • Firefox • Safari
Outlook on the web (classic)	• iPhone 4 or later • iPad 2 or later • iPod Touch 4 or later	• iOS 5 or later	On on-premises Exchange Server 2016 or later[2]	• Safari
Outlook on Android	• Android tablets • Android smartphones	• Android 4.4 KitKat or later	On the latest update of Microsoft 365 Apps for business or Exchange Online	Browser not applicable. Use the native app for Android.[3]
Outlook on iOS	• iPad tablets • iPhone smartphones	• iOS 11 or later • iPadOS 13 or later	On the latest update of Microsoft 365 Apps for business or Exchange Online	Browser not applicable. Use the native app for iOS.[3]

Note

[1] Modern Outlook on the web on iPhone and Android smartphones is no longer required or available for testing Outlook add-ins.

[2] Add-ins aren't supported in Outlook on Android, on iOS, and modern mobile web with on-premises Exchange accounts.

[3] OWA for Android, OWA for iPad, and OWA for iPhone native apps have been deprecated.

Tip

You can distinguish between classic and modern Outlook in a web browser by checking your mailbox toolbar.

modern

classic

See also

- Office Add-ins platform overview
- Office client application and platform availability for Office Add-ins
- Browsers and webview controls used by Office Add-ins

Install the latest version of Office

Article • 09/24/2024

New developer features, including those still in preview, are delivered first to subscribers who opt in to get the latest builds of Office.

Opt in to getting the latest builds of Office

- If you're a Microsoft 365 Family, Personal, or University subscriber, see Be a Microsoft 365 Insider.
- If you're a Microsoft 365 Apps for business customer, see Microsoft 365 Insider for Business.
- If you're running Office on a Mac:
  - Start an Office application.
  - Select **Check for Updates** on the Help menu.
  - In the Microsoft AutoUpdate box, check the box to join the Microsoft 365 Insider program.

Get the latest build of Office

1. Download the Office Deployment Tool.

2. Run the tool. This extracts a **setup.exe** and configuration files.

3. Create a new file named **configuration.xml** and add the following XML.

```xml
<Configuration>
  <Add OfficeClientEdition="32" Branch="CurrentPreview">
    <Product ID="O365ProPlusRetail">
      <Language ID="en-us" />
    </Product>
  </Add>

  <Updates Enabled="TRUE" />
  <Display Level="None" AcceptEULA="TRUE" />

</Configuration>
```

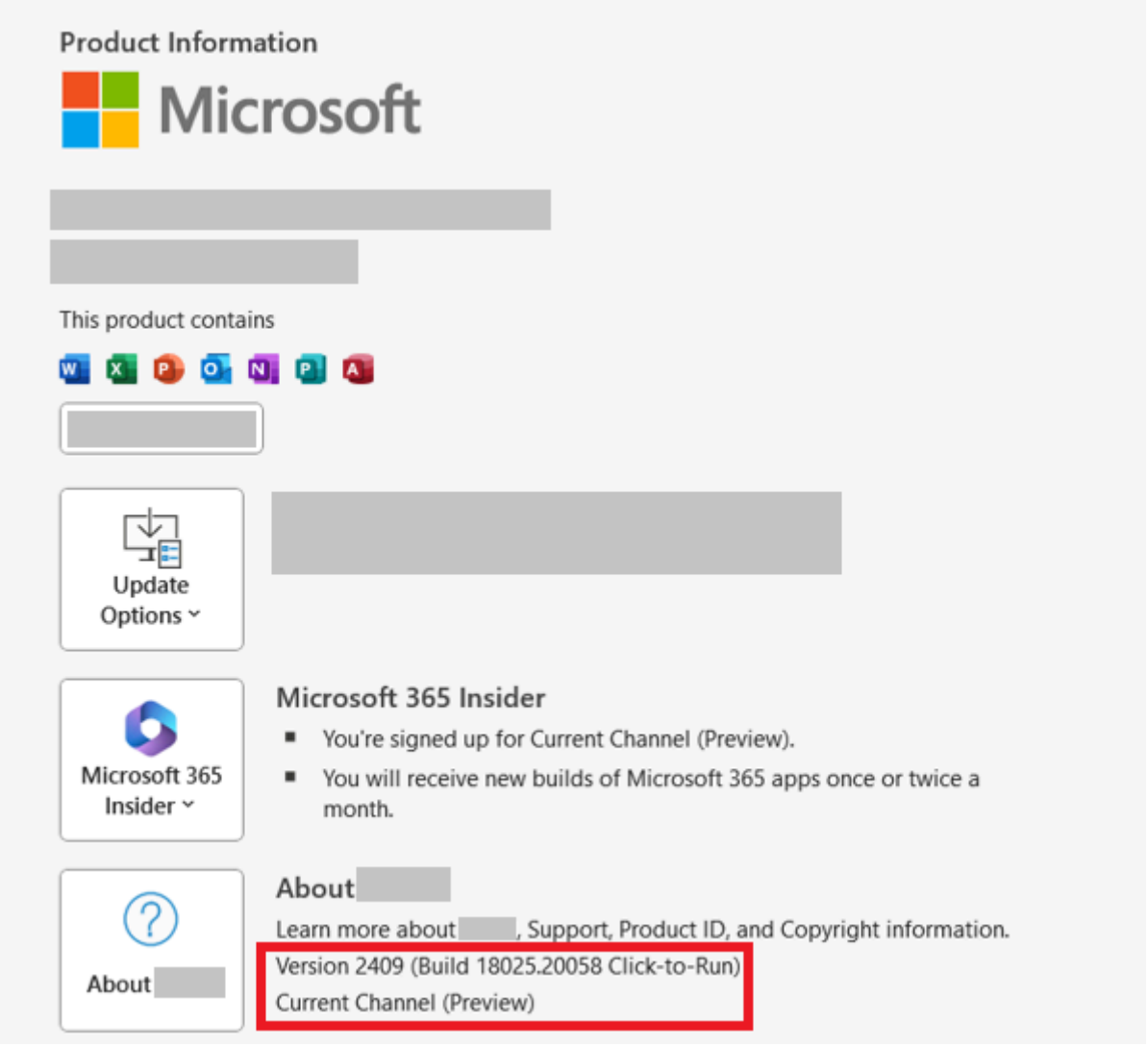
4. Run the following command as an administrator.

```
command line
```

```
setup.exe /configure configuration.xml
```

> **Note**
>
> The command might take a long time to run without indicating progress.

When the installation process finishes, you'll have the latest Office applications installed. To verify that you have the latest build, go to **File** > **Account** from any Office application. Under the About section, you'll see the version and build number, along with Current Channel (Preview). The Microsoft 365 Insider section is displayed or hidden for business customers depending on their company's settings.



Minimum Office builds for Office JavaScript API requirement sets

- Excel JavaScript API requirement sets

- OneNote JavaScript API requirement sets
- Outlook JavaScript API requirement sets
- PowerPoint JavaScript API requirement sets
- Word JavaScript API requirement sets
- Dialog API requirement sets
- Office Common API requirement sets

Browsers and webview controls used by Office Add-ins

Article • 10/17/2024

Office Add-ins are web applications that are displayed using iframes when running in Office on the web. In Office for desktop and mobile clients, Office Add-ins use an embedded browser control (also known as a webview). Add-ins also need a JavaScript engine to run the JavaScript. Both the embedded browser and the engine are supplied by a browser installed on the user's computer. In this article, "webview" refers to the combination of a webview control and a JavaScript engine.

Which webview is used depends on:

- The computer's operating system.
- Whether the add-in is running in Office on the web, in Office downloaded from a Microsoft 365 subscription, or in perpetual Office 2016 or later.
- Within the perpetual versions of Office on Windows, whether the add-in is running in the "retail" or "volume-licensed" variation.

> ⓘ **Important**
>
> **Webviews from Internet Explorer and Microsoft Edge Legacy are still used in Office Add-ins**
>
> Some combinations of platforms and Office versions, including volume-licensed perpetual versions through Office 2019, still use the webview controls that come with Internet Explorer 11 (called "Trident") and Microsoft Edge Legacy (called "EdgeHTML") to host add-ins, as explained in this article. Internet Explorer 11 was disabled in Windows 10 and Windows 11 in February 2023, and the UI for launching it was removed; but it's still installed on those operating systems. So, Trident and other functionality from Internet Explorer can still be called programmatically by Office.
>
> We recommend (but don't require) that you continue to support these combinations, at least in a minimal way, by providing users of your add-in a graceful failure message when your add-in is launched in one of these webviews. Keep these additional points in mind:
>
> - Office on the web no longer opens in Internet Explorer or Microsoft Edge Legacy. Consequently, AppSource doesn't test add-ins in Office on these web

browsers.

- AppSource still tests for combinations of platform and Office *desktop* versions that use Trident or EdgeHTML. However, it only issues a warning when the add-in doesn't support these webviews; the add-in isn't rejected by AppSource.
- The Script Lab tool no longer supports Trident.

For more information about supporting Trident or EdgeHTML, including configuring a graceful failure message on your add-in, see Support older Microsoft webviews and Office versions.

The following sections specify which browser is used for the various platforms and operating systems.

Non-Windows platforms

For these platforms, the platform alone determines the browser that's used.

Expand table

OS	Office version	Browser
any	Office on the web	The browser in which Office is opened. (But note that Office on the web will not open in Internet Explorer. Attempting to do so opens Office on the web in Edge.)
Mac	any	Safari with WKWebView
iOS	any	Safari with WKWebView
Android	any	Chrome

Important

Conditional Access isn't supported for Office Add-ins on iOS or Android. Those add-ins use the Safari-based WKWebView or the Android-based WebView, not an Edge-based browser control.

Windows

An add-in running on Windows might use any of three different webviews:

- **WebView2**, which is provided by Microsoft Edge (Chromium-based).
- **EdgeHTML**, which is provided by Microsoft Edge Legacy.
- **Trident+**, which is provided by Internet Explorer 11. The "+" on the end indicates that Office Add-ins use additional functionality from Internet Explorer 11 that isn't built into Trident itself.

Perpetual versions of Office on Windows

For perpetual versions of Office on Windows, the browser that's used is determined by the Office version, whether the license is retail or volume-licensed, and whether the Edge WebView2 (Chromium-based) is installed. The version of Windows doesn't matter, but note that Office Add-ins aren't supported on versions earlier than Windows 7 and Office 2021 and later aren't supported on versions earlier than Windows 10.

To determine whether Office 2016 or Office 2019 is retail or volume-licensed, use the format of the Office version and build number. (For Office 2021 and later, the distinction between volume-licensed and retail doesn't matter.)

- **Retail**: For both Office 2016 and 2019, the format is `YYMM (xxxxx.xxxxxx)`, ending with two blocks of five digits; for example, `2206 (Build 15330.20264)`.
- **Volume-licensed**:
  - For Office 2016, the format is `16.0.xxxx.xxxxx`, ending with two blocks of *four* digits; for example, `16.0.5197.1000`.
  - For Office 2019, the format is `1808 (xxxxx.xxxxxx)`, ending with two blocks of *five* digits; for example, `1808 (Build 10388.20027)`. Note that the year and month is always `1808`.

Expand table

Office version	Retail vs. Volume-licensed	WebView2 installed?	Browser
Office 2024	Doesn't matter	Yes[1]	WebView2 (Microsoft Edge[2] Chromium-based)
Office 2021	Doesn't matter	Yes[1]	WebView2 (Microsoft Edge[2] Chromium-based)
Office 2019	Retail	Yes[1]	WebView2 (Microsoft Edge[2] Chromium-based)
Office 2019	Retail	No	EdgeHTML (Microsoft Edge Legacy)[2, 3] If Edge isn't installed, Trident+ (Internet Explorer 11) is used.

Office version	Retail vs. Volume-licensed	WebView2 installed?	Browser
Office 2019	Volume-licensed	Doesn't matter	Trident+ (Internet Explorer 11)
Office 2016	Retail	Yes[1]	WebView2 (Microsoft Edge[2] Chromium-based)
Office 2016	Retail	No	EdgeHTML (Microsoft Edge Legacy)[2, 3] If Edge isn't installed, Trident+ (Internet Explorer 11) is used.
Office 2016	Volume-licensed	Doesn't matter	Trident+ (Internet Explorer 11)

[1] On Windows versions prior to Windows 11, the WebView2 control must be installed so that Office can embed it. It's installed with perpetual Office 2021 or later; but it isn't automatically installed with Microsoft Edge. If you have an earlier version of perpetual Office, use the instructions for installing the control at Microsoft Edge WebView2 / Embed web content ... with Microsoft Edge WebView2.

[2] When you use either EdgeHTML or WebView2, the Windows Narrator (sometimes called a "screen reader") reads the `<title>` tag in the page that opens in the task pane. In Trident+, the Narrator reads the title bar of the task pane, which comes from the add-in name that's specified in the add-in's manifest.

[3] If your add-in uses an add-in only manifest and includes the **<Runtimes>** element in the manifest or it uses the unified manifest and it includes an "extensions.runtimes.lifetime" property, then it won't use EdgeHTML. If the conditions for using WebView2 are met, then the add-in uses WebView2. Otherwise, it uses Trident+. For more information, see Runtimes and Configure your Outlook add-in for event-based activation.

Microsoft 365 subscription versions of Office on Windows

For subscription Office on Windows, the browser that's used is determined by the operating system, the Office version, and whether the WebView2 control is installed.

Expand table

OS	Office version	WebView2 installed?	Browser
• Windows 11 • Windows 10 • Windows 8.1	Microsoft 365 ver. >= 16.0.13530.20424[1]	Yes[2]	WebView2 (Microsoft Edge[3]

OS	Office version	WebView2 installed?	Browser
• Windows Server 2022 • Windows Server 2019 • Windows Server 2016			Chromium-based)
• Window 11 • Windows 10 ver. >= 1903	Microsoft 365 ver. >= 16.0.13530.20424[1]	No	EdgeHTML (Microsoft Edge Legacy)[3, 4]
• Windows 11 • Windows 10 ver. >= 1903	Microsoft 365 ver. >= 16.0.11629 *AND* < 16.0.13530.20424[1]	Doesn't matter	EdgeHTML (Microsoft Edge Legacy)[3, 4]
• Windows 11 • Windows 10 ver. >= 1903	Microsoft 365 ver. < 16.0.11629[1]	Doesn't matter	Trident+ (Internet Explorer 11)
• Windows 10 ver. < 1903 • Windows 8.1	Microsoft 365	No	Trident+ (Internet Explorer 11)
• Windows 7	Microsoft 365	Doesn't matter	Trident+ (Internet Explorer 11)

[1] See the update history page and how to find your Office client version and update channel for more details.

[2] On Windows versions prior to Windows 11, the WebView2 control must be installed so that Office can embed it. It's installed with Microsoft 365, Version 2101 or later, but it isn't automatically installed with Microsoft Edge. If you have an earlier version of Microsoft 365, use the instructions for installing the control at Microsoft Edge WebView2 / Embed web content ... with Microsoft Edge WebView2. On Microsoft 365 builds before 16.0.14326.xxxxx, you must also create the registry key **HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\Win32WebView2** and set its value to `dword:00000001`.

[3] When you use either EdgeHTML or WebView2, the Windows Narrator (sometimes called a "screen reader") reads the `<title>` tag in the page that opens in the task pane.

In Trident+, the Narrator reads the title bar of the task pane, which comes from the add-in name that's specified in the add-in's manifest.

[4] If your add-in uses an add-in only manifest and includes the **<Runtimes>** element in the manifest or it uses the unified manifest and it includes an "extensions.runtimes.lifetime" property, then it won't use EdgeHTML. If the conditions for using WebView2 are met, then the add-in uses WebView2. Otherwise, it uses Trident+. For more information, see Runtimes and Configure your Outlook add-in for event-based activation.

Working with Trident+ (Internet Explorer 11)

Trident+ doesn't support JavaScript versions later than ES5. If any of your add-in's users have platforms that use Trident+, then to use the syntax and features of ECMAScript 2015 or later, you have two options.

- Write your code in ECMAScript 2015 (also called ES6) or later JavaScript, or in TypeScript, and then compile your code to ES5 JavaScript using a compiler such as babel or tsc.
- Write in ECMAScript 2015 or later JavaScript, but also load a polyfill library such as core-js that enables IE to run your code.

For more information about these options, see Support older Microsoft webviews and Office versions.

Also, Trident+ doesn't support some HTML5 features such as media, recording, and location. To learn more, see Determine the webview the add-in is running in at runtime.

Troubleshoot EdgeHTML and WebView2 (Microsoft Edge) issues

Service Workers aren't working

Office Add-ins don't support Service Workers when EdgeHTML is used. They're supported with WebView2.

Scroll bar doesn't appear in task pane

By default, scroll bars in EdgeHTML and WebView2 are hidden until hovered over. To ensure that the scroll bar is always visible, the CSS styling that applies to the `<body>`

element of the pages in the task pane should include the -ms-overflow-style property and it should be set to `scrollbar`.

When debugging with the Microsoft Edge DevTools, the add-in crashes or reloads

Setting breakpoints in the Microsoft Edge DevTools for EdgeHTML can cause Office to think that the add-in is hung. It will automatically reload the add-in when this happens. To prevent this, add the following Registry key and value to the development computer: `[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\16.0\Wef]"AlertInterval"=dword:00000000`.

When the add-in tries to open, get "ADD-IN ERROR We can't open this add-in from the localhost" error

One known cause is that EdgeHTML requires that localhost be given a loopback exemption on the development computer. Follow the instructions at Cannot open add-in from localhost.

Get errors trying to download a PDF file

Directly downloading blobs as PDF files in an add-in isn't supported with EdgeHTML or WebView2. The workaround is to create a simple web application that downloads blobs as PDF files. In your add-in, call the `Office.context.ui.openBrowserWindow(url)` method and pass the URL of the web application. This will open the web application in a browser window outside of Office.

WIP-protected documents

There's an extra step needed for Add-ins to run in a document with WIP (Windows Information Protection) and use **WebView2 (Microsoft Edge Chromium-based)**. Add the WebView2 process, **msedgewebview2.exe**, to the protected apps list in your enterprise's WIP policy. An admin adds this WIP policy through Intune with the following values.

- **Name**: Webview2
- **Publisher**: O=MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US
- **Product Name**: MICROSOFT EDGE WEBVIEW2
- **File**: MSEDGEWEBVIEW2.EXE
- **Min Version**: *

- **Max Version**: *

If the WIP policy hasn't been added, the add-in defaults to an older runtime. In the sections Perpetual versions of Office on Windows and Microsoft 365 subscription versions of Office on Windows earlier in this article, substitute **EdgeHTML (Microsoft Edge Legacy)** for **WebView2 (Microsoft Edge Chromium-based)** wherever the latter appears.

To determine if a document is WIP-protected, follow these steps.

1. Open the file.
2. Select the **File** tab on the ribbon.
3. Select **Info**.
4. In the upper section of the **Info** page, just below the file name, a WIP-enabled document will have a briefcase icon followed by **Managed by Work (...)**.

> ⓘ **Note**
>
> Support for WebView2 in WIP-enabled documents was added with build 16.0.16626.20132. If you're on an older build, your runtime defaults to **EdgeHTML (Microsoft Edge Legacy)**, regardless of policy.

See also

- Requirements for Running Office Add-ins
- Runtimes in Office Add-ins