Office Add-ins glossary

Article • 02/12/2025

This is a glossary of terms commonly used throughout the Office Add-ins documentation.

add-in

Office Add-ins are web applications that extend Office applications. These web applications add new functionality to the Office application, such as bring in external data, automate processes, or embed interactive objects in Office documents.

Office Add-ins differ from VBA, COM, and VSTO add-ins because they offer cross-platform support (usually web, Windows, Mac, and iPad) and are based on standard web technologies (HTML, CSS, and JavaScript). The primary programming language of an Office Add-in is JavaScript or TypeScript.

add-in commands

Add-in commands are UI elements, such as buttons and menus, that extend the Office UI for your add-in. When users select an add-in command element, they initiate actions such as running JavaScript code or displaying the add-in in a task pane. Add-in commands let your add-in look and feel like a part of Office, which gives users more confidence in your add-in. See Add-in commands to learn more.

See also: ribbon, ribbon button.

application

Application refers to an Office application. The Office applications that support Office Add-ins are Excel, OneNote, Outlook, PowerPoint, Project, and Word.

See also: client, host, Office application, Office client.

application-specific API

Application-specific APIs provide strongly-typed objects that interact with objects that are native to a specific Office application. For example, you call the Excel JavaScript APIs for access to worksheets, ranges, tables, charts, and more. Application-specific APIs are

currently available for Excel, OneNote, PowerPoint, Visio, and Word. See Application-specific API model to learn more.

See also: Common API.

client

Client typically refers to an Office application. The Office applications, or clients, that support Office Add-ins are Excel, OneNote, Outlook, PowerPoint, Project, and Word.

See also: application, host, Office application, Office client, Office desktop application, Office desktop client, desktop client, desktop.

Common API

Common APIs are used to access features such as UI, dialogs, and client settings that are common across multiple Office applications. This API model uses callbacks, which allow you to specify only one operation in each request sent to the Office application.

Common APIs were introduced with Office 2013. Some Common APIs are legacy APIs from the early 2010s. Excel, PowerPoint, and Word all have Common API functionality, but most of this functionality has been replaced or superseded by the application-specific API model. The application-specific APIs are preferred when possible.

Other Common APIs, such as the Common APIs related to Outlook, UI, and authentication, are the modern and preferred APIs for these purposes. For details about the Common API object model, see Common JavaScript API object model.

See also: application-specific API.

content add-in

Content add-ins are webviews, or web browser views, that are embedded directly into Excel, OneNote, or PowerPoint documents. Content add-ins give users access to interface controls that run code to modify documents or display data from a data source. Use content add-ins when you want to embed functionality directly into the document. See Content Office Add-ins to learn more.

See also: webview.

content delivery network (CDN)

A **content delivery network** or **CDN** is a distributed network of servers and data centers. It typically provides higher resource availability and performance when compared to a single server or data center.

Contoso

Contoso Ltd. (also known as Contoso and Contoso University) is a fictional company used by Microsoft as an example company and domain.

custom function

A **custom function** is a user-defined function that is packaged with an Excel add-in. Custom functions enable developers to add new functions, beyond the typical Excel features, by defining those functions in JavaScript as part of an add-in. Users within Excel can access custom functions just as they would any native function in Excel. See Create custom functions in Excel to learn more.

> ⓘ Note
>
> **Custom function** is a general term that is interchangeable with **user-defined function**. Both terms apply to VBA, COM, and Office.js add-ins. The Office Add-ins documentation uses the term **custom function** when referring to custom functions that use Office JavaScript APIs.

custom functions runtime

A **custom functions runtime** is a JavaScript-only runtime that runs custom functions on some combinations of Office host and platform. It has no UI and cannot interact with Office.js APIs. If your add-in only has custom functions, this is a good lightweight runtime to use. If your custom functions need to interact with the task pane or Office.js APIs, configure a shared runtime. See Configure your Office Add-in to use a shared runtime to learn more.

See also: runtime, shared runtime.

custom functions-only add-in

An add-in that contains a custom function, but no UI such as a task pane. The custom functions in this kind of add-in run in a JavaScript-only runtime. A custom function that

does include a UI can use either a shared runtime or a combination of a JavaScript-only runtime and an HTML-supporting runtime. We recommend that if you have a UI, you use a shared runtime.

See also: custom function, custom functions runtime.

function command

Function commands are buttons or menu items that run JavaScript functions. Unlike task pane commands, function commands don't display any user interface other than the command button or menu item itself.

See also: add-in commands.

host

<Host> typically refers to an Office application. The Office applications, or hosts, that support Office Add-ins are Excel, OneNote, Outlook, PowerPoint, Project, and Word.

See also: application, client, Office application, Office client.

Long-Term Service Channel (LTSC)

LTSC refers to the perpetual version of Office available through a volume-licensing agreement between Microsoft and your company.

See also: perpetual, volume-licensed, volume-licensed perpetual, volume licensing.

Office application, Office client

Office client refers to an Office application. The Office applications, or clients, that support Office Add-ins are Excel, OneNote, Outlook, PowerPoint, Project, and Word.

See also: application, client, host, Office desktop application, Office desktop client, desktop client, desktop.

Office cache

The **Office cache** stores resources and data used by Office Add-ins. This cache prevents an add-in from repeatedly downloding the resources it needs, thereby improving its performance.

See also: web cache, Wef cache.

Office desktop application, Office desktop client, desktop client, desktop

Office desktop client refers to an Office application that runs natively on Windows or on Mac. The Office desktop clients that support Office Add-ins are Excel on Windows and on Mac, Outlook on Windows (new and classic) and on Mac, PowerPoint on Windows and on Mac, Project on Windows, and Word on Windows and on Mac.

See also: application, client, Office application, Office client.

perpetual

Perpetual refers to versions of Office available through a volume-licensing agreement or retail channels.

Other Microsoft content may use the term **non-subscription** to represent this concept.

See also: retail, retail perpetual, volume-licensed, volume-licensed perpetual, volume licensing.

platform

A **platform** usually refers to the operating system running the Office application. Platforms that support Office Add-ins include Windows, Mac, iPad, and web browsers.

quick start

A **quick start** is a high-level description of key skills and knowledge required for the basic operation of a particular program. In the Office Add-ins documentation, a quick start is an introduction to developing an add-in for a particular application, such as Outlook. A quick start contains a series of steps that an add-in developer can complete in approximately 5 minutes, resulting in a functioning add-in and functional development environment.

See also: tutorial.

requirement set

Requirement sets are named groups of API members. Requirement sets can be specific to Office applications, such as the `ExcelApi 1.7` requirement set (a set of APIs that can only be used in Excel), or common to multiple applications, such as the `DialogApi 1.1` requirement set (a set of APIs that can be used in any Office application that supports the Dialog API).

Your add-in can use requirement sets to determine whether the Office application supports the API members that it needs to use. For more information about this, see Specify Office applications and API requirements.

Requirement set support varies by Office application, version, and platform. For detailed information about the platforms, requirement sets, and Common APIs that each Office application supports, see Office client application and platform availability for Office Add-ins.

retail, retail perpetual

Retail refers to perpetual versions of Office available through retail channels. These do not include versions provided by a Microsoft 365 subscription nor volume-licensing agreement.

Other Microsoft content may use the term **one-time purchase** or **consumer** to represent this concept.

See also: perpetual.

ribbon, ribbon button

A **ribbon** is a command bar that organizes an application's features into a series of tabs or buttons at the top of a window. A **ribbon button** is one of the buttons within this series. See Show or hide the ribbon in Office for more information.

runtime

A **runtime** is the host environment (including a JavaScript engine and usually also an HTML rendering engine) that the add-in runs in. In Office on Windows and Office on Mac, the runtime is an embedded browser control (or webview) such as Internet Explorer, Edge Legacy, Edge WebView2, or Safari. Different parts of an add-in run in separate runtimes. For example, add-in commands, custom functions, and task pane code typically use separate runtimes unless you configure a shared runtime. See

Runtimes in Office Add-ins and Browsers and webview controls used by Office Add-ins for more information.

See also: custom functions runtime, shared runtime, webview.

setless API

An API in the Office JavaScript Library that is not included in any requirement set.

See also requirement set.

shared runtime

A **shared runtime**, enables code in your task pane, function commands, and custom functions, to run in the same runtime and continue running even when the task pane is closed. Code in dialogs generally runs in a separate runtime even when the add-in is configured to use a shared runtime. See shared runtime and Tips for using the shared runtime in your Office Add-in to learn more.

See also: custom functions runtime, runtime.

subscription

Subscription refers to versions of Office available with a Microsoft 365 subscription.

task pane

Task panes are interface surfaces, or webviews, that typically appear on the right side of the window within Excel, Outlook, PowerPoint, and Word. Task panes give users access to interface controls that run code to modify documents or emails, or display data from a data source. Use task panes when you don't need to or can't embed functionality directly into the document. See Task panes in Office Add-ins to learn more.

See also: webview.

tutorial

A **tutorial** is a teaching aid designed to help people learn to use a product or procedure. In the Office Add-ins context, a tutorial guides an add-in developer through the

complete add-in development process for a particular application, such as Excel. This involves following 20 or more steps and is a greater time investment than a quick start.

See also: quick start.

volume-licensed, volume-licensed perpetual, volume licensing

Volume-licensed refers to a perpetual version of Office available through a volume-licensing agreement between Microsoft and your company.

Other Microsoft content may use the term **commercial** to represent this concept.

See also: Long-Term Service Channel (LTSC), perpetual.

web add-in

Web add-in is a legacy term for an Office Add-in. This term may be used when the Microsoft 365 documentation needs to distinguish modern Office Add-ins from other types of add-ins like VBA, COM, or VSTO.

See also: add-in.

web cache

The **web cache** temporarily stores web-based resources and data used by an individual Office Add-in.

See also: Office cache, Wef cache.

webview

A **webview** is an element or view that displays web content inside an application. Content add-ins and task panes both contain embedded web browsers and are examples of webviews in Office Add-ins.

See also: content add-in, task pane.

Wef cache

The **Wef cache** locally stores resources and data for all installed Office Add-ins.

See also: Office cache, web cache.

XLL

An **XLL** add-in is an Excel add-in file that provides user-defined functions and has the file extension **.xll**. An XLL file is a type of dynamic link library (DLL) file that can only be opened by Excel. XLL add-in files must be written in C or C++. Custom functions are the modern equivalent of XLL user-defined functions. Custom functions offer support across platforms and are backwards compatible with XLL files. See Extend custom functions with XLL user-defined functions for more information.

See also: custom function.

Yeoman generator, Yo Office

The Yeoman generator for Office Add-ins uses the open source Yeoman tool to generate an Office Add-in via the command line. `yo office` is the command that runs the Yeoman generator for Office Add-ins. The Office Add-ins quick starts and tutorials use the Yeoman generator.

See also

- Office Add-ins additional resources