200 XP

Understand testing, debugging, and deployment options for Office add-ins

7 minutes

In this unit, you'll explore testing, debugging, and deployment options for Office add-ins. By the end of this unit, you should know how to sideload your add-in to test it, how to use Visual Studio, Visual Studio Code, or the browser developer tools to debug your add-in, and the different options available for deployment.

Choose the best deployment option for your Office add-in

As you develop your Office add-in and prepare to make it available to your users, you need to decide which deployment option is best. The following table lists factors you should consider.

Consider...	Examples
Add-in lifecycle stage	local developer testing, ready for public use
Add-in interaction or feature support	task pane add-in, content add-in, add-in commands
Target Office applications	Excel, Outlook
Target platforms	Windows, Mac
Scope of user base	your organization, general public

Deployment options

You have several options for deploying your add-in. The following table notes each option and when it should be used.

Option	Description	Best when...
Sideload	Install your add-in locally.	Developer building and testing add-in

Option	Description	Best when...
Centralized deployment	Distribute your add-in to users via the Microsoft 365 admin center.	Add-in ready for use in your organization on Office 365 or in a hybrid environment
SharePoint catalog	Distribute add-in to users via SharePoint.	Task pane or content add-in ready for use in your organization that's using an on-premises environment; Excel, Word, or PowerPoint is targeted but Mac isn't a target platform
AppSource	Make add-in available to the public.	Add-in ready for public use
Exchange server	Distribute add-in to users via Exchange.	Outlook add-in ready for use in an organization whose environment doesn't use Azure Active Directory identity service
Network share	Make add-in available to network users via a shared folder.	Add-in development and users are on Windows

Understand testing and debugging concepts for Office add-ins

At various points during your add-in's life cycle, you need to verify functionality and fix bugs. You have several options for how you go about testing and debugging your add-in.

Sideload your add-in

You can locally install (sideload) your add-in for testing and debugging on Windows, Mac, and in a web browser. You can also sideload your Excel or Word add-in on an iPad. Use Node.js, Internet Information Services (IIS), or another preferred means to web host your add-in on your development machine.

If you create your project using the Yeoman generator for Office add-ins, you can run **npm run start** in a command-line prompt to start and sideload your add-in to Excel on Windows or **npm run start:web** to run it in a web browser, though you'll have to manually sideload to Excel in the browser.

If you create your project using Visual Studio (VS), you can run the project in VS debug mode and it will automatically sideload to Excel on Windows.

Debug your add-in

You can debug your add-in using the following methods:

- A web browser with the browser's built-in developer tools
- Visual Studio, provided you prepared your add-in using this IDE
- Visual Studio Code for custom functions projects only
- Runtime logging on Windows and Mac

If you need to debug your add-in on a specific platform, there are more tools that may help you. A few options for Windows and Mac are mentioned next in this section.

Windows

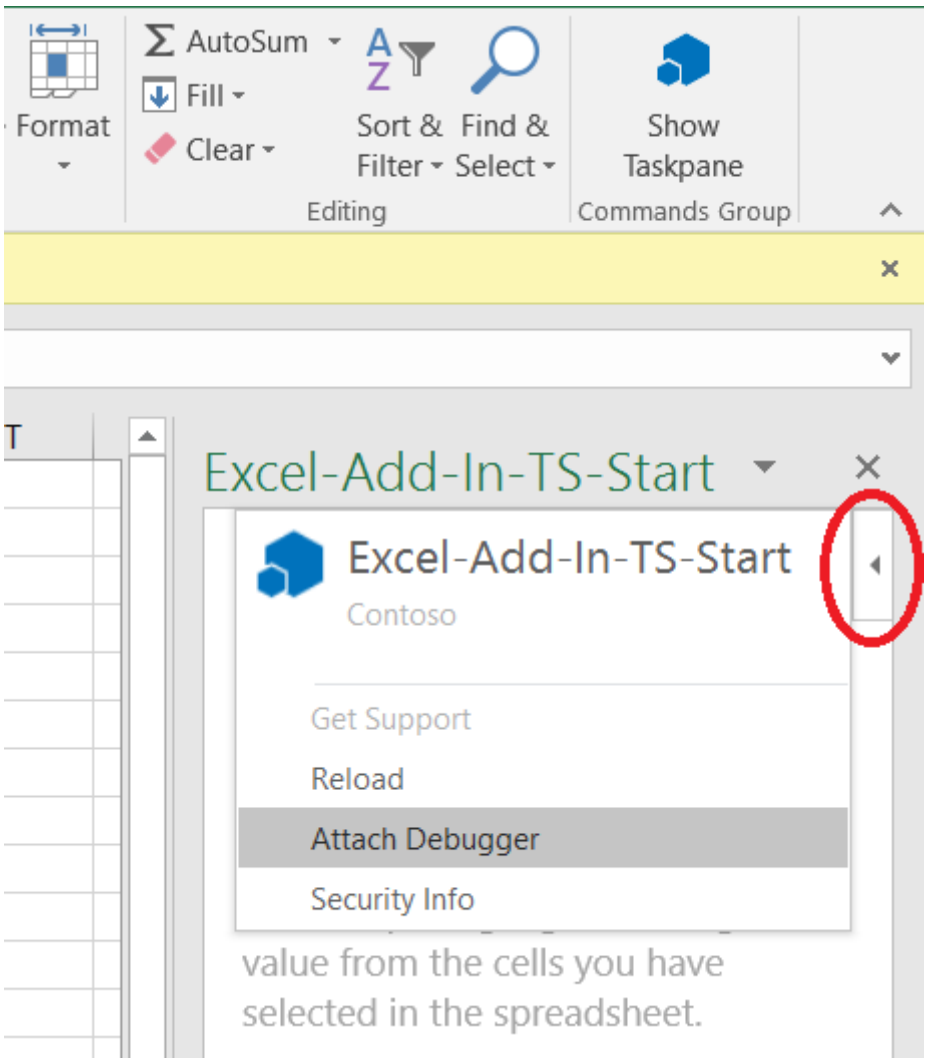
On Windows 10, the tool you use depends on if the add-in is running in Microsoft Edge or Internet Explorer. Your add-in is running in Internet Explorer 11 unless it meets the following criteria to be running in Microsoft Edge.

- Windows 10 (version 1903 or later)
- Office 365 subscription (build 16.0.11629 or later)

For Microsoft Edge, install and use Microsoft Edge DevTools. For Internet Explorer, run F12 developer tools according to your Office version:

- 32-bit Office version: **C:\Windows\System32\F12\IEChooser.exe**
- 64-bit Office version: **C:\Windows\SysWOW64\F12\IEChooser.exe**

An available option to debug task pane add-ins in Office 2016 or later is to attach a debugger. Where the **Attach Debugger** is available through the Personality menu, as shown in the following image, the supported tool is Visual Studio 2015 Update 3 or later. This tool only enables JavaScript debugging.



*Personality menu displaying **Attach Debugger** item*

If the Personality menu isn't present or you're already using Visual Studio (VS), you can use **Attach to Process** in VS to debug the add-in in Microsoft Edge or Internet Explorer as applicable.

Mac

For your sideloaded task pane and content add-ins, you can use the Safari Web Inspector on macOS High Sierra and Office version 16.9.1 (build 18012504) or later. The supported Office applications are:

- Excel
- Outlook
- PowerPoint
- Word

Validate your manifest

You can validate your add-in's manifest using any of these options:

- Yeoman generator for Office add-ins
- office-addin-manifest **validate** command
- libxml

Test required Office clients and platforms

Test your add-in in Office versions and on platforms where you or your intended users will be using it.

Private use or limited to your organization

If your add-in will be limited to you or your organization, test it in Office versions and on platforms where they'll use it. For example, if you're developing a Word add-in for your organization where your coworkers usually work in Microsoft Edge and Word 2019 on Windows, test your add-in in that browser and version of Word.

Public use

If your add-in will be available to the public through AppSource, you should look over the AppSource validation policies so review and validation of your add-in is as smooth as possible. A few key validation requirements are:

- Browsers: Internet Explorer 11 and later, Microsoft Edge, Chrome, Firefox, and Safari (Mac)
- Office: All applications you specified in the `Hosts` section of the add-in's manifest configuration file
- Operating systems: Windows, Mac, and iPad—if your Outlook add-in supports mobile, include iOS and Android

AppSource validation policy 4.12 discusses the expected client and platform support requirements in more detail.

Summary

In this unit, you explored testing, debugging, and deployment options for Office add-ins. You should now know how to sideload your add-in to test it, how to use Visual Studio, Visual Studio Code, or the browser developer tools to debug your add-in, and the different options available for deployment.

Testing, debugging, and deployment options for Office add-ins

1. A developer is about to test and debug their new add-in. What's the best deployment option?

- ○ AppSource
- ○ Exchange server
- ○ Sideloading

2. To debug a custom function, a developer should use which tool?

- ○ Visual Studio
- ○ Visual Studio Code
- ○ Yeoman generator for Office add-ins

3. A developer would like to publish their add-in on AppSource. What are the three (3) key areas they should validate?

- ○ Browsers, Office applications, Operating systems
- ○ Browsers, Office applications, Organizations
- ○ Monitor resolutions, Office applications, Operating systems

Check your answers