

Debug your add-in with runtime logging

Article • 11/12/2024

You can use runtime logging to debug your add-in's manifest as well as several installation errors. This feature can help you identify and fix issues with your manifest that are not detected by XSD schema validation, such as a mismatch between resource IDs. Runtime logging is particularly useful for debugging add-ins that implement add-in commands and Excel custom functions.

ⓘ Note

The runtime logging feature is currently available for Office 2016 or later on desktop.

ⓘ Important

Runtime Logging affects performance. Turn it on only when you need to debug issues with your add-in manifest.

Use runtime logging from the command line

Enabling runtime logging from the command line is the fastest way to use this logging tool.

ⓘ Important

The office-addin-dev-settings tool is not supported on Mac. See the section [Runtime logging on Mac](#) for Mac-specific instructions.

- To enable runtime logging:

```
command line
```

```
npx office-addin-dev-settings runtime-log --enable
```

- To enable runtime logging only for a specific file, use the same command with a filename:

command line

```
npx office-addin-dev-settings runtime-log --enable [filename.txt]
```

- To disable runtime logging:

command line

```
npx office-addin-dev-settings runtime-log --disable
```

- To display whether runtime logging is enabled:

command line

```
npx office-addin-dev-settings runtime-log
```

- To display help within the command line for runtime logging:

command line

```
npx office-addin-dev-settings runtime-log --help
```

Runtime logging on Mac

1. Make sure that you are running Office 2016 desktop build **16.27.19071500** or later.
2. Open **Terminal** and set a runtime logging preference by using the `defaults` command:

command line

```
defaults write <bundle id> CEFRuntimeLoggingFile -string <file_name>
```

`<bundle id>` identifies which the host for which to enable runtime logging.

`<file_name>` is the name of the text file to which the log will be written.

Set `<bundle id>` to one of the following values to enable runtime logging for the corresponding application.

- `com.microsoft.Word`
- `com.microsoft.Excel`
- `com.microsoft.Powerpoint`

- `com.microsoft.Outlook`

The following example enables runtime logging for Word and then opens the log file.

command line

```
defaults write com.microsoft.Word CEFRuntimeLoggingFile -string  
"runtime_logs.txt"  
open ~/library/Containers/com.microsoft.Word/Data/runtime_logs.txt
```

ⓘ Note

You'll need to restart Office after running the `defaults` command to enable runtime logging.

To turn off runtime logging, use the `defaults delete` command:

command line

```
defaults delete <bundle id> CEFRuntimeLoggingFile
```

The following example will turn off runtime logging for Word.

command line

```
defaults delete com.microsoft.Word CEFRuntimeLoggingFile
```

Use runtime logging to troubleshoot issues with your manifest

To use runtime logging to troubleshoot issues loading an add-in:

1. [Sideload your add-in](#) for testing.

ⓘ Note

We recommend that you sideload only the add-in that you are testing to minimize the number of messages in the log file.

2. If nothing happens and you don't see your add-in (and it's not appearing in the add-ins dialog box), open the log file.

3. Search the log file for your add-in ID, which you define in your manifest. In the log file, this ID is labeled `SolutionId`.

Known issues with runtime logging

You might see messages in the log file that are confusing or that are classified incorrectly. For example:

- The message `Medium Current host not in add-in's host list` followed by `Unexpected Parsed manifest targeting different host` is incorrectly classified as an error.
- If you see the message `Unexpected Add-in is missing required manifest fields DisplayName` and it doesn't contain a `SolutionId`, the error is most likely not related to the add-in you are debugging.
- Any `Monitorable` messages are expected errors from a system point of view. Sometimes they indicate an issue with your manifest, such as a misspelled element that was skipped but didn't cause the manifest to fail.

See also

- [Office Add-ins manifest](#)
- [Validate an Office Add-in's manifest](#)
- [Clear the Office cache](#)
- [Sideload Office Add-ins for testing](#)
- [Debug add-ins using developer tools for Internet Explorer](#)
- [Debug add-ins using developer tools for Edge Legacy](#)
- [Debug add-ins using developer tools in Microsoft Edge \(Chromium-based\)](#)
- [Runtimes in Office Add-ins](#)

Debug a function command with a non-shared runtime

06/17/2025


Important

If your add-in is [configured to use a shared runtime](#), debug the code behind the function command just as you would the code behind a task pane. See [Debug Office Add-ins](#) and note that a function command in an add-in with a [shared runtime](#) is *not* a special case as described in that article.

Note

This article assumes that you're familiar with [function commands](#).

Function commands don't have a UI, so a debugger can't be attached to the process in which the function runs on desktop Office. (Outlook add-ins being developed on Windows are an exception to this. See [Debug function commands in Outlook add-ins on Windows](#) later in this article.) So function commands, in add-ins with a non-shared runtime, must be debugged in Office on the web where the function runs in the overall browser process. Use the following steps.

1. [Sideload the add-in in Office on the web](#), and then select the button or menu item that runs the function command. This is necessary to load the code file for the function command.
2. Open the browser's developer tools. This is usually done by pressing . The debugger in the tools attaches to the browser process.
3. Apply breakpoints to the code as needed for the function command.
4. Rerun the function command. The process stops on your breakpoints.

Tip

For more detailed information, see [Debug add-ins in Office on the web](#).

Debug function commands in Outlook add-ins on Windows

If your development computer is Windows, there's a way that you can debug a function command on Outlook desktop. See [Debug function commands in Outlook add-ins](#).

See also

- [Runtimes in Office Add-ins](#)

Debug the initialize and onReady functions

06/17/2025

ⓘ Note

This article assumes that you're familiar with [Initialize your Office Add-in](#).

The paradox of debugging the `Office.initialize` and `Office.onReady` functions is that a debugger can only attach to a process that's running, but these functions run immediately as the add-in's runtime process starts up, before a debugger can attach. In most situations, restarting the add-in after a debugger is attached doesn't help because restarting the add-in closes the original runtime process *and the attached debugger* and starts a new process that has no debugger attached.

Fortunately, there's an exception. You can debug these functions using Office on the web, with the following steps.

1. Sideload and run the add-in in Office on the web. This is usually done by opening an add-in's task pane or running a [function command](#). *The add-in runs in the overall browser process, not a separate process as it would in desktop Office.*
2. Open the browser's developer tools. This is usually done by pressing `F12`. The debugger in the tools attaches to the browser process.
3. Apply breakpoints as needed to the code in the `Office.initialize` or `Office.onReady` function.
4. *Relaunch the add-in's task pane or the function command* just as you did in step 1. This action does *not* close the browser process or the debugger. The `Office.initialize` or `Office.onReady` function runs again and processing stops on your breakpoints.

💡 Tip

For more detailed information, see [Debug add-ins in Office on the web](#).

See also

- [Runtimes in Office Add-ins](#)

Error handling with the application-specific JavaScript APIs

06/23/2025

When you build an add-in using the [application-specific Office JavaScript APIs](#), be sure to include error handling logic to account for runtime errors. Doing so is critical, due to the asynchronous nature of the APIs.

Best practices

In our [code samples](#) and [Script Lab](#) snippets, you'll notice that every call to `Excel.run`, `PowerPoint.run`, or `Word.run` is accompanied by a `catch` statement to catch any errors. We recommend that you use the same pattern when you build an add-in using the application-specific APIs.

JavaScript

```
$("#run").on("click", () => tryCatch(run));

async function run() {
  await Excel.run(async (context) => {
    // Add your Excel JavaScript API calls here.

    // Await the completion of context.sync() before continuing.
    await context.sync();
    console.log("Finished!");
  });
}

/** Default helper for invoking an action and handling errors. */
async function tryCatch(callback) {
  try {
    await callback();
  } catch (error) {
    // Note: In a production add-in, you'd want to notify the user through your
    add-in's UI.
    console.error(error);
  }
}
```

API errors

When an Office JavaScript API request doesn't run successfully, the API returns an error object that contains the following properties.

- **code:** The `code` property of an error message contains a string that is part of `OfficeExtension.ErrorCodes` or `{application}.ErrorCodes` where *{application}* represents Excel, PowerPoint, or Word. For example, the error code "InvalidReference" indicates that the reference is not valid for the specified operation. Error codes are not localized.
- **message:** The `message` property of an error message contains a summary of the error in the localized string. The error message isn't intended for consumption by end users; you should use the error code and appropriate business logic to determine the error message that your add-in shows to end users.
- **debugInfo:** When present, the `debugInfo` property of the error message provides additional information that you can use to understand the root cause of the error.

ⓘ Note


If you use `console.log()` to print error messages to the console, those messages are only visible on the server. End users don't see those error messages in the add-in task pane or anywhere in the Office application. To report errors to the user, see [Error notifications](#).

Error codes and messages

The following tables list the errors that application-specific APIs may return.

ⓘ Note

The following tables list error messages you may encounter while using the application-specific APIs. If you're working with the Common API, see [Office Common API error codes](#) to learn about relevant error messages.

 Expand table


Error code	Error message	Notes
<code>AccessDenied</code>	You cannot perform the requested operation.	This may be caused by a user's antivirus software blocking parts of Office. See the Common errors and troubleshooting steps for "Error: Access denied" for more guidance.

Error code	Error message	Notes
ActivityLimitReached	Activity limit has been reached.	None
ApiNotAvailable	The requested API is not available.	None
ApiNotFound	The API you are trying to use could not be found. It may be available in a newer version of the Office application. See Office client application and platform availability for Office Add-ins for more information.	None
BadPassword	The password you supplied is incorrect.	None
Conflict	Request could not be processed because of a conflict.	None
ContentLengthRequired	A Content-length HTTP header is missing.	None
GeneralException	There was an internal error while processing the request.	None
HostRestartNeeded	The Office application needs to be restarted.	This error is thrown by the Office.ribbon.requestUpdate() method if the add-in that calls the method has been updated since the Office application started.
InsertDeleteConflict	The insert or delete operation attempted resulted in a conflict.	None
InvalidArgument	The argument is invalid or missing or has an incorrect format.	None
InvalidBinding	This object binding is no longer valid due to previous updates.	None

Error code	Error message	Notes
<code>InvalidOperation</code>	The operation attempted is invalid on the object.	<i>None</i>
<code>InvalidReference</code>	This reference is not valid for the current operation.	<i>None</i>
<code>InvalidRequest</code>	Cannot process the request.	<i>None</i>
<code>InvalidRibbonDefinition</code>	Office has been given an invalid ribbon definition.	This error is thrown if an invalid RibbonUpdateObject is passed to the <code>Office.ribbon.requestUpdate()</code> method.
<code>InvalidSelection</code>	The current selection is invalid for this operation.	<i>None</i>
<code>ItemAlreadyExists</code>	The resource being created already exists.	<i>None</i>
<code>ItemNotFound</code>	The requested resource doesn't exist.	<i>None</i>
<code>MemoryLimitReached</code>	The memory limit has been reached. Your action could not be completed.	<i>None</i>
<code>NotImplemented</code>	The requested feature isn't implemented.	This could mean the API is in preview or only supported on a particular platform (such as online-only). See Office client application and platform availability for Office Add-ins for more information.
<code>RequestAborted</code>	The request was aborted during run time.	<i>None</i>
<code>RequestPayloadSizeLimitExceeded</code>	The request payload size has exceeded the limit. See the Resource limits and performance optimization for Office Add-ins article for more information.	This error only occurs in Office on the web.
<code>ResponsePayloadSizeLimitExceeded</code>	The response payload size has exceeded the limit. See the Resource limits and performance optimization for Office	This error only occurs in Office on the web.

Error code	Error message	Notes
	Add-ins article for more information.	
ServiceNotAvailable	The service is unavailable.	None
Unauthenticated	Required authentication information is either missing or invalid.	None
UnsupportedFeature	The operation failed because the source worksheet contains one or more unsupported features.	None
UnsupportedOperation	The operation being attempted is not supported.	None

Excel-specific error codes and messages

 Expand table

Error code	Error message	Notes
EmptyChartSeries	The attempted operation failed because the chart series is empty.	None
FilteredRangeConflict	The attempted operation causes a conflict with a filtered range.	None
FormulaLengthExceedsLimit	The bytecode of the applied formula exceeds the maximum length limit. For Office on 32-bit machines, the bytecode length limit is 16384 characters. On 64-bit machines, the bytecode length limit is 32768 characters.	This error occurs in both Excel on the web and on desktop.

Error code	Error message	Notes
<code>GeneralException</code>	<i>Various.</i>	The data types APIs return <code>GeneralException</code> errors with dynamic error messages. These messages reference the cell that is the source of the error, and the problem that is causing the error, such as: "Cell A1 is missing the required property <code>type</code> ."
<code>InactiveWorkbook</code>	The operation failed because multiple workbooks are open and the workbook being called by this API has lost focus.	<i>None</i>
<code>InvalidOperationInCellEditMode</code>	The operation isn't available while Excel is in Edit cell mode. Exit Edit mode by using the <code>Enter</code> or <code>Tab</code> keys, or by selecting another cell, and then try again.	<i>None</i>
<code>MergedRangeConflict</code>	Cannot complete the operation. A table can't overlap with another table, a PivotTable report, query results, merged cells, or an XML Map.	<i>None</i>
<code>NonBlankCellOffSheet</code>	Microsoft Excel can't insert new cells because it would push non-empty cells off the end of the worksheet. These non-empty cells might appear empty but have blank values, some formatting, or a formula. Delete enough rows or columns to make room for what you want to insert and then try again.	<i>None</i>
<code>OperationCellsExceedLimit</code>	The attempted operation affects more	If the <code>TableColumnCollection.add</code> API triggers this error, confirm that there is no

Error code	Error message	Notes
	than the limit of 33554000 cells.	<p>unintentional data within the worksheet but outside of the table. In particular, check for data in the right-most columns of the worksheet. Remove the unintended data to resolve this error. One way to verify how many cells that an operation processes is to run the following calculation: $(\text{number of table rows}) \times (16383 - (\text{number of table columns}))$. The number 16383 is the maximum number of columns that Excel supports.</p> <p>This error only occurs in Excel on the web.</p>
PivotTableRangeConflict	The attempted operation causes a conflict with a PivotTable range.	None
RangeExceedsLimit	<p>The cell count in the range has exceeded the maximum supported number.</p> <p>See the Resource limits and performance optimization for Office Add-ins article for more information.</p>	None
RefreshWorkbookLinksBlocked	The operation failed because the user hasn't granted permission to refresh external workbook links.	None
UndoNotSupported	The JavaScript API request failed due to lack of support for the undo operation.	None
UnsupportedSheet	This sheet type does not support this operation, since it is a Macro or Chart sheet.	None

Word-specific error codes and messages

Error code	Error message	Notes
<code>SearchDialogIsOpen</code>	The search dialog is open.	<i>None</i>
<code>SearchStringInvalidOrTooLong</code>	The search string is invalid or too long.	The search string maximum is 255 characters.

Error notifications

How you report errors to users depends on the UI system you're using.

- If you're using React as the UI system, use the [Fluent UI](#) components and design elements. We recommend that error messages be conveyed with a [Dialog](#) component. If the error is in the user's input, configure the [Input](#) component to display the error as bold red text.

ⓘ Note

The [Alert](#) component can also be used to report errors to users, but it's currently in preview and shouldn't be used in a production add-in. For information about its release status, see the [Fluent UI React v9 Component Roadmap](#).

- If you're not using React for the UI, consider using the older [Fabric UI](#) components implemented directly in HTML and JavaScript. Some example templates are in the [Office-Add-in-UX-Design-Patterns-Code](#) repository. Take a look especially in the dialog and navigation subfolders. The sample [Excel-Add-in-SalesLeads](#) uses a message banner.

See also

- [OfficeExtension.Error](#) object
- [Office Common API error codes](#)

Office Common API error codes

Article • 12/27/2022

This article documents the error messages you might encounter while using the Common API model. These error codes don't apply to application-specific APIs, such as the Excel JavaScript API or the Word JavaScript API.

See [API models](#) to learn more about the differences between the Common API and application-specific API models.

Error codes

The following table lists the error codes, names, and messages displayed, and the conditions they indicate.

 Expand table

Error.code	Error.name	Error.message	Condition
1000	Invalid Coercion Type	The specified coercion type is not supported	The coercion type is not supported in the Office application. (For example, OOXML and HTML coercion types are not supported in Excel.)
1001	Data Read Error	The current selection is not supported.	The user's current selection is not supported (that is, it is something different than the supported coercion types).
1002	Invalid Coercion Type	The specified coercion type is not compatible for this binding type.	The solution developer provided an incompatible combination of coercion type and binding type.
1003	Data Read Error	The specified rowCount or columnCount values are invalid.	The user supplies invalid column or row counts.
1004	Data Read Error	The current selection is not compatible for the specified coercion type.	The current selection is not supported for the specified coercion type by this application.
1005	Data Read Error	The specified startRow or	The user supplies invalid startRow or startCol values.

Error.code	Error.name	Error.message	Condition
		startColumn values are invalid.	
1006	Data Read Error	Coordinate parameters cannot be used with coercion type "Table" when the table contains merged cells.	The user tries to get partial data from a non-uniform table (that is, a table that has merged cells).
1007	Data Read Error	The size of the document is too large.	The user tries to get a document larger than the size currently supported.
1008	Data Read Error	The requested data set is too large.	The user requests to read data beyond the data limits defined by the Office application.
1009	Data Read Error	The specified file type is not supported.	The user sends an invalid file type.
2000	Data Write Error	The supplied data object type is not supported.	An unsupported data object is supplied.
2001	Data Write Error	Cannot write to the current selection.	The user's current selection is not supported for a write operation. (For example, when the user selects an image.)
2002	Data Write Error	The supplied data object is not compatible with the shape or dimensions of the current selection.	Multiple cells are selected (and the selection shape does not match the shape of the data). Multiple cells are selected (and the selection dimensions do not match the dimensions of the data).
2003	Data Write Error	The set operation failed because the supplied data object will overwrite data.	A single cell is selected and the supplied data object overwrites data in the worksheet.
2004	Data Write Error	The supplied data object does not match the size of the current selection.	The user supplies an object larger than the current selection size.

Error.code	Error.name	Error.message	Condition
2005	Data Write Error	The specified startRow or startColumn values are invalid.	The user supplies invalid startRow or startCol values.
2006	Invalid Format Error	The format of the specified data object is not valid.	The solution developer supplies an invalid HTML or OOXML string, a malformed HTML string, or an invalid OOXML string.
2007	Invalid Data Object	The type of the specified data object is not compatible with the current selection.	The solution developer supplies a data object not compatible with the specified coercion type.
2008	Data Write Error	TBD	TBD
2009	Data Write Error	The specified data object is too large.	The user tries to set data beyond the data limits defined by the Office application.
2010	Data Write Error	Coordinate parameters cannot be used with coercion type Table when the table contains merged cells.	The user tries to set partial data from a non-uniform table (that is, a table that has merged cells).
3000	Binding Creation Error	Cannot bind to the current selection.	The user's selection is not supported for binding. (For example, the user is selecting an image or other non-supported object.)
3001	Binding Creation Error	TBD	TBD
3002	Invalid Binding Error	The specified binding does not exist.	The developer tries to bind to a non-existing or removed binding.
3003	Binding Creation Error	Noncontiguous selections are not supported.	The user is making multiple selections.
3004	Binding Creation Error	A binding cannot be created with the current selection	There are several conditions under which this might happen. Please see the "Binding

Error.code	Error.name	Error.message	Condition
		and the specified binding type.	creation error conditions" section later in this article.
3005	Invalid Binding Operation	Operation is not supported on this binding type.	The developer sends an add row or add column operation on a binding type that is not of coercion type <code>table</code> .
3006	Binding Creation Error	The named item does not exist.	The named item cannot be found. No content control or table with that name exists.
3007	Binding Creation Error	Multiple objects with the same name were found.	Collision error: more than one content control with the same name exists, and fail on collision is set to <code>true</code> .
3008	Binding Creation Error	The specified binding type is not compatible with the supplied named item.	Named item can't be bound to type. For example, a content control contains text, but the developer tried to bind by using coercion type <code>table</code> .
3009	Invalid Binding Operation	The binding type is not supported.	Used for backward compatibility.
3010	Unsupported Binding Operation	The selected content needs to be in table format. Format the data as a table and try again.	The developer is trying to use the <code>addRowsAsync</code> or <code>deleteAllDataValuesAsync</code> method of the <code>TableBinding</code> object on data of coercion type <code>matrix</code> .
4000	Read Settings Error	The specified setting name does not exist.	A nonexistent setting name is supplied.
4001	Save Settings Error	The settings could not be saved.	Settings could not be saved.
4002	Settings Stale Error	Settings could not be saved because they are stale.	Settings are stale and developer indicated not to override settings.
5000	Settings Stale Error	The operation is not supported.	The operation is not supported in the current Office application. For example, <code>document.getSelectionAsync</code> is called from Outlook.

Error.code	Error.name	Error.message	Condition															
5001	Internal Error	An internal error has occurred.	<div>Refers to an internal error condition, which can occur for any of the following reasons.</div> <div><div><div></div></div><div>Expand table</div></div> <table><tr><td>An add-in being used by another user sharing the workbook created a binding at approximately the same time, and your add-in needs to retry binding.</td></tr><tr><td>An unknown error occurred.</td></tr><tr><td>The operation failed.</td></tr><tr><td>Access was denied because the user is not a member of an authorized role.</td></tr><tr><td>Access was denied because secure, encrypted communication is required.</td></tr><tr><td>Data is stale and the user needs to confirm enabling the queries to refresh it.</td></tr><tr><td>The site collection CPU quota has been exceeded.</td></tr><tr><td>The site collection memory quota has been exceeded.</td></tr><tr><td>The session memory quota has been exceeded.</td></tr><tr><td>The workbook is in an invalid state and the operation can't be performed.</td></tr><tr><td>The session has timed out due to inactivity and the user needs to reload the workbook.</td></tr><tr><td>The maximum number of allowed sessions per user has been exceeded.</td></tr><tr><td>The operation was canceled by the user.</td></tr><tr><td>The operation can't be completed because it is taking too long.</td></tr><tr><td>The request can't be completed and needs to be retried.</td></tr></table>	An add-in being used by another user sharing the workbook created a binding at approximately the same time, and your add-in needs to retry binding.	An unknown error occurred.	The operation failed.	Access was denied because the user is not a member of an authorized role.	Access was denied because secure, encrypted communication is required.	Data is stale and the user needs to confirm enabling the queries to refresh it.	The site collection CPU quota has been exceeded.	The site collection memory quota has been exceeded.	The session memory quota has been exceeded.	The workbook is in an invalid state and the operation can't be performed.	The session has timed out due to inactivity and the user needs to reload the workbook.	The maximum number of allowed sessions per user has been exceeded.	The operation was canceled by the user.	The operation can't be completed because it is taking too long.	The request can't be completed and needs to be retried.
An add-in being used by another user sharing the workbook created a binding at approximately the same time, and your add-in needs to retry binding.																		
An unknown error occurred.																		
The operation failed.																		
Access was denied because the user is not a member of an authorized role.																		
Access was denied because secure, encrypted communication is required.																		
Data is stale and the user needs to confirm enabling the queries to refresh it.																		
The site collection CPU quota has been exceeded.																		
The site collection memory quota has been exceeded.																		
The session memory quota has been exceeded.																		
The workbook is in an invalid state and the operation can't be performed.																		
The session has timed out due to inactivity and the user needs to reload the workbook.																		
The maximum number of allowed sessions per user has been exceeded.																		
The operation was canceled by the user.																		
The operation can't be completed because it is taking too long.																		
The request can't be completed and needs to be retried.																		

Error.code	Error.name	Error.message	Condition
			The trial period of the product has expired.
			The session has timed out due to inactivity.
			The user doesn't have permission to perform the operation on the specified range.
			The user's regional settings don't match the current collaboration session.
			The user is no longer connected and must refresh or re-open the workbook.
			The requested range doesn't exist in the sheet.
			The user doesn't have permission to edit the workbook.
			The workbook can't be edited because it is locked.
			The session can't auto save the workbook.
			The session can't refresh its lock on the workbook file.
			The request can't be processed and needs to be retried.
			The user's sign-in information couldn't be verified and needs to be re-entered.
			The user has been denied access.
			The shared workbook needs to be updated.
5002	Permission Denied	The requested operation is not allowed on the current document mode.	The solution developer submits a set operation, but the document is in a mode that does not allow modifications, such as 'Restrict Editing'.
5003	Event Registration Error	The specified event type is not supported by the current object.	The solution developer tries to register or unregister a handler to an event that does not exist.

Error.code	Error.name	Error.message	Condition
5004	Invalid API call	Invalid API call in the current context.	An invalid call is made for the context, for example, trying to use a <code>CustomXMLPart</code> object in Excel.
5005	Data Stale	Operation failed because the data is stale on the server.	The data on the server needs to be refreshed.
5006	Session Timeout	The document session timed out. Reload the document.	The session has timed out.
5007	Invalid API call	The enumeration is not supported in the current context.	The enumeration is not supported in the current context.
5009	Permission Denied	Access Denied	The add-in does not have permission to call the specific API.
5012	Invalid Or Timed Out Session	Your Office browser session has expired or is invalid. To continue, refresh the page.	The session between the Office client and server has expired or the date, time, or time zone is incorrect on your computer.
6000	Invalid node	The specified node was not found.	The <code>CustomXmlPart</code> node was not found.
6100	Custom XML error	Custom XML error	Invalid API call.
7000	Invalid Id	The specified Id does not exist.	Invalid ID.
7001	Invalid navigation	The object is located in a place where navigation is not supported.	The user can find the object, but cannot navigate to it. (For example, in Word, the binding is to the header, footer, or a comment.)
7002	Invalid navigation	The object is locked or protected.	The user is trying to navigate to a locked or protected range.
7004	Invalid navigation	The operation failed because the Index is out of range.	The user is trying to navigate to an index that is out of range.

Error.code	Error.name	Error.message	Condition
8000	Missing Parameter	We couldn't format the table cell because some parameter values are missing. Double-check the parameters and try again.	The cellFormat method is missing some parameters. For example, there are missing cells, format, or tableOptions parameters.
8010	Invalid value	One or more of the cells parameters have values that aren't allowed. Double-check the values and try again.	The common cells reference enumeration is not defined. For example, All, Data, Headers.
8011	Invalid value	One or more of the tableOptions parameters have values that aren't allowed. Double-check the values and try again.	One of the values in tableOptions is invalid.
8012	Invalid value	One or more of the format parameters have values that aren't allowed. Double-check the values and try again.	One of the values in the format is invalid.
8020	Out of range	The row index value is out of the allowed range. Use a positive value (0 or higher) that's less than the number of rows.	The row index is more than the biggest row index of the table or less than 0.
8021	Out of range	The column index value is out of the allowed range. Use a positive value (0 or higher) that's less than the	The column index is more than the biggest column index of the table or less than 0.

Error.code	Error.name	Error.message	Condition
		number of columns.	
8022	Out of range	The value is out of the allowed range.	Some of the values in the format are out of the supported ranges.
9016	Permission denied	Permission denied	Access is denied.
9020	Generic Response Error	An internal error has occurred.	Refers to an internal error condition, which can occur for any number of reasons.
9021	Save Error	Connection error occurred while trying to save the item on the server.	The item couldn't be saved. This could be due to a server connection error if using Online Mode in Outlook desktop, or due to an attempt to re-save a draft item that was deleted from the Exchange server.
9022	Message In Different Store Error	The EWS ID cannot be retrieved because the message is saved in another store.	The EWS ID for the current message couldn't be retrieved as the message may have been moved or the sending mailbox may have changed.
9041	Network error	The user is no longer connected to the network. Please check your network connection and try again.	The user no longer has network or internet access.
9043	Attachment Type Not Supported	The attachment type is not supported.	The API doesn't support the attachment type. For example, <code>item.getAttachmentContentAsync</code> throws this error if the attachment is an embedded image in Rich Text Format, or if it's an item type other than an email or calendar item (such as a contact or task item).
9057	Size Limit Exceeded	A maximum of 32KB is available for the settings of each add-in.	When updating roaming settings via <code>Office.context.roamingSettings.set</code> , the size cannot exceed 32KB. See Office.RoamingSettings interface .
12002	<i>Not applicable</i>	<i>Not applicable</i>	One of the following: - No page exists at the URL that was passed to <code>displayDialogAsync</code> . - The page that was passed to

Error.code	Error.name	Error.message	Condition
			<code>displayDialogAsync</code> loaded, but the dialog box was directed to a page that it cannot find or load, or it has been directed to a URL with invalid syntax. Thrown within the dialog and triggers a <code>DialogEventReceived</code> event in the host page.
12003	<i>Not applicable</i>	<i>Not applicable</i>	The dialog box was directed to a URL with the HTTP protocol. HTTPS is required. Thrown within the dialog and triggers a <code>DialogEventReceived</code> event in the host page.
12004	<i>Not applicable</i>	<i>Not applicable</i>	The domain of the URL passed to <code>displayDialogAsync</code> is not trusted. The domain must be the same domain as the host page (including protocol and port number). Thrown by call of <code>displayDialogAsync</code> .
12005	<i>Not applicable</i>	<i>Not applicable</i>	The URL passed to <code>displayDialogAsync</code> uses the HTTP protocol. HTTPS is required. Thrown by call of <code>displayDialogAsync</code> . (In some versions of Office, the error message returned with 12005 is the same one returned for 12004.)
12006	<i>Not applicable</i>	<i>Not applicable</i>	The dialog box was closed, usually because the user chooses the X button. Thrown within the dialog and triggers a <code>DialogEventReceived</code> event in the host page.
12007	<i>Not applicable</i>	<i>Not applicable</i>	A dialog box is already opened from this host window. A host window, such as a task pane, can only have one dialog box open at a time. Thrown by call of <code>displayDialogAsync</code> .
12009	<i>Not applicable</i>	<i>Not applicable</i>	The user chose to ignore the dialog box. This error can occur in online versions of Office, where users may choose not to allow an add-in to present a dialog. Thrown by call of <code>displayDialogAsync</code> .
12011	<i>Not applicable</i>	<i>Not applicable</i>	The user's browser is configured in a way that blocks popups. This error can occur in Office on the web if the browser is Safari and it's configured to block popups or the browser is Edge Legacy and the add-in domain is in a different security zone from the domain the

Error.code	Error.name	Error.message	Condition
			dialog is trying to open. Thrown by call of <code>displayDialogAsync</code> .
13nnn	<i>Not applicable</i>	<i>Not applicable</i>	See Causes and handling of errors from getAccessToken .

Binding creation error conditions

When a binding is created in the API, indicate the binding type that you want to use. The following tables lists the binding types and the resulting binding behaviors that are expected.

Behavior in Excel

The following table summarizes binding behavior in Excel.

 Expand table

Specified Binding Type	Actual Selection	Behavior
Matrix	Range of cells (including within a table, and single cell)	A binding of type <code>matrix</code> is created on the selected cells. No modification in the document is expected.
Matrix	Text selected in the cell	A binding of type <code>matrix</code> is created on the whole cell. No modification in the document is expected.
Matrix	Multiple selection/invalid selection (For example, user selects a picture, object, or Word Art.)	The binding cannot be created.
Table	Range of cells (includes single cell)	The binding cannot be created.
Table	Range of cell within a table (includes single cell within a table, or the whole table, or text within a cell in a table)	A binding is created in the whole table.
Table	Half selection in a table and half selection outside the table	The binding cannot be created.
Table	Text selected in the cell (not in the table.)	The binding cannot be created.

Specified Binding Type	Actual Selection	Behavior
Table	Multiple selection/invalid selection (For example, user selects a picture, object, Word Art, etc.)	The binding cannot be created.
Text	Range of cells	The binding cannot be created.
Text	Range of cells within a table	The binding cannot be created.
Text	Single cell	A binding of type <code>text</code> is created.
Text	Single cell within a table	A binding of type <code>text</code> is created.
Text	Text selected in the cell	A binding of type <code>text</code> in the whole cell is created.

Behavior in Word

The following table summarizes binding behavior in Word.

[Expand table](#)

Specified Binding Type	Actual Selection	Behavior
Matrix	Text	The binding cannot be created.
Matrix	Whole table	A binding of type <code>matrix</code> is created.Document is changed and a content control must wrap the table.
Matrix	Range within a table	The binding cannot be created.
Matrix	Invalid selection (for example, multiple, invalid objects, etc.)	The binding cannot be created.
Table	Text	The binding cannot be created.
Table	Whole table	A binding of type <code>text</code> is created.
Table	Range within a table	The binding cannot be created.
Table	Invalid selection (for example, multiple, invalid objects, etc.)	The binding cannot be created.

Specified Binding Type	Actual Selection	Behavior
Text	Whole table	A binding of type <code>text</code> is created.
Text	Range within a table	The binding cannot be created.
Text	Multiple selection	The last selection will be wrapped with a content control and a binding to that control. A content control of type <code>text</code> is created.
Text	Invalid selection (for example, multiple, invalid objects, etc.)	The binding cannot be created.

See also

- [Office Add-ins development lifecycle](#)
- [Understanding the Office JavaScript API](#)
- [Error handling with the application-specific JavaScript APIs](#)
- [Troubleshoot error messages for single sign-on \(SSO\)](#)
- [Troubleshoot development errors with Office Add-ins](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

Office Add-ins feedback

Office Add-ins is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Troubleshoot user errors with Office Add-ins

06/25/2025

At times your users might encounter issues with Office Add-ins that you develop. For example, an add-in fails to load or is inaccessible. Use the information in this article to help resolve common issues that your users encounter with your Office Add-in.

You can also use tools to intercept HTTP messages to identify and debug issues with your add-ins. Popular choices include [Fiddler](#), [Charles](#), and [Requestly](#).

Common errors and troubleshooting steps

The following table lists common error messages that users might encounter and steps that your users can take to resolve the errors.

 Expand table

Error message	Resolution
App error: Catalog could not be reached	Verify firewall settings."Catalog" refers to AppSource. This message indicates that the user cannot access AppSource.
APP ERROR: This app could not be started. Close this dialog to ignore the problem or click "Restart" to try again.	Verify that the latest Office updates are installed, or update with the Windows Installer .
Error: Access denied. E_ACCESSDENIED (0x80070005)	The antivirus software installed on the machine might prevent the host app from creating a WebView2 process. To resolve this issue, add an exemption or exclusion to the antivirus for the .exe files in the Office root folder (C:\Program Files\Microsoft Office\root\Office16) or for the entire Office root folder. If this does not fix the issue, add an exemption or exclusion for the WebView2 process (C:\Program Files (x86)\Microsoft\EdgeWebView\Application[latest installed version]\msedgewebview2.exe).
Error: Object doesn't support property or method 'defineProperty'	Confirm that Internet Explorer is not running in Compatibility Mode. Go to Tools > Compatibility View Settings .
Sorry, we couldn't load the app because your	Make sure that the browser supports HTML5 local storage, or reset your Internet Explorer settings. For information about supported browsers, see

Error message	Resolution
browser version is not supported. Click here for a list of supported browser versions.	Requirements for running Office Add-ins.

When installing an add-in, you see "Error loading add-ins" in the status bar

1. Close Office.
2. Check that the time and date are set correctly on your computer. An incorrect time and date can cause issues when verifying the add-in's manifest.
3. Verify that the manifest is valid. See [Validate an Office Add-in's manifest](#).
4. Restart the add-in.
5. Install the add-in again.

If the add-in package was tampered with before installation, this error will occur. Download the add-in again and try to reinstall it. Alternatively, contact the publisher of the add-in for help.

You can also give us feedback: if using Office on Windows or Mac, you can report feedback to the Office extensibility team directly from Office. To do this, select **Help > Feedback > Report a problem**. Sending a report provides necessary information to understand the issue.

Outlook add-in doesn't work correctly

If an Outlook add-in running on Windows and [using Internet Explorer](#) is not working correctly, try turning on script debugging in Internet Explorer.

- Go to **Tools > Internet Options > Advanced**.
- Under **Browsing**, uncheck **Disable script debugging (Internet Explorer)** and **Disable script debugging (Other)**.

We recommend that you uncheck these settings only to troubleshoot the issue. If you leave them unchecked, you will get prompts when you browse. After the issue is resolved, check **Disable script debugging (Internet Explorer)** and **Disable script debugging (Other)** again.

Add-in doesn't activate in Office

If the add-in doesn't activate when the user performs the following steps.

1. Signs in with their Microsoft account in the Office application.

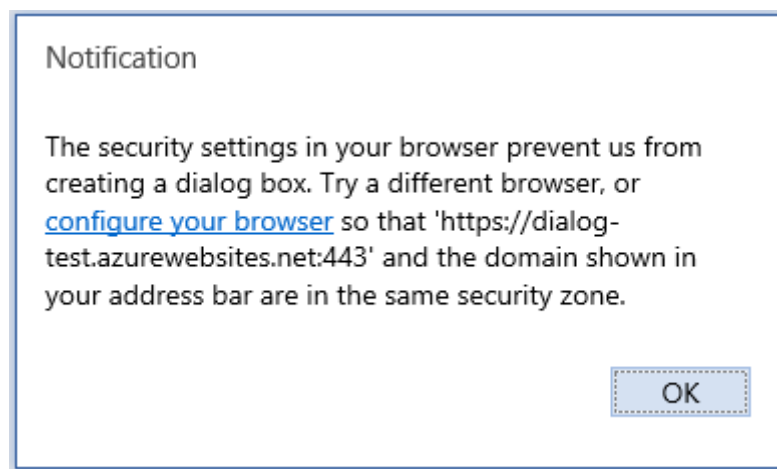
2. Enables two-step verification for their Microsoft account.
3. Verifies their identity when prompted when they try to insert an add-in.

Verify that the latest Office updates are installed, or update with the [Windows Installer](#).

Add-in dialog box cannot be displayed

When using an Office Add-in, the user is asked to allow a dialog box to be displayed. The user chooses **Allow**, and the following error message occurs.

"The security settings in your browser prevent us from creating a dialog box. Try a different browser, or configure your browser so that [URL] and the domain shown in your address bar are in the same security zone."



[Expand table](#)

Affected browsers	Affected platforms
Microsoft Edge	Office on the web

To resolve the issue, end users or administrators can add the domain of the add-in to the list of trusted sites in the Microsoft Edge browser.

Important

Do not add the URL for an add-in to your list of trusted sites if you don't trust the add-in.

To add a URL to your list of trusted sites:

1. In **Control Panel**, go to **Internet options > Security**.
2. Select the **Trusted sites** zone, and choose **Sites**.

3. Enter the URL that appears in the error message, and choose **Add**.
4. Try to use the add-in again. If the problem persists, verify the settings for the other security zones and ensure that the add-in domain is in the same zone as the URL that is displayed in the address bar of the Office application.

This issue occurs when the Dialog API is used in pop-up mode. To prevent this issue from occurring, use the [displayInFrame](#) flag. This requires that your page support display within an iframe. The following example shows how to use the flag.

JavaScript

```
Office.context.ui.displayDialogAsync(startAddress, {displayInIFrame:true},  
callback);
```

Add-in won't upgrade

You may see the following error when deploying an updated manifest for your add-in: **ADD-IN**

WARNING: This add-in is currently upgrading. Please close the current message or appointment, and re-open in a few moments.

When you add features or fix bugs in your add-in, you'll need to deploy the updates. If your add-in is deployed by one or more admins to their organizations, some manifest changes will require the admin to consent to the updates. Users remain on the existing version of the add-in until the admin consents to the updates. The following manifest changes will require the admin to consent again.

- Changes to requested permissions. See [Requesting permissions for API use in add-ins](#) and [Understanding Outlook add-in permissions](#).
- Additional or changed [Scopes](#). (Not applicable if the add-in uses the unified manifest for Microsoft 365.)
- Additional or changed [Outlook events](#).

ⓘ Note

Whenever you make a change to the manifest, you must raise the version number of the manifest.

- If the add-in uses the add-in only manifest, see [Version element](#).
- If the add-in uses the unified manifest, see [version property](#).

See also

- [Troubleshoot development errors with Office Add-ins](#)

Troubleshoot development errors with Office Add-ins

Article • 02/12/2025

Here's a list of common issues you may encounter while developing an Office Add-in.

Tip

Clearing the Office cache often fixes issues related to stale code. This guarantees the latest manifest is uploaded, using the current file names, menu text, and other command elements. To learn more, see [Clear the Office cache](#).

Add-in doesn't load in task pane or other issues with the add-in manifest

See [Validate an Office Add-in's manifest](#) and [Debug your add-in with runtime logging](#) to debug add-in manifest issues.

Ribbon customizations are not rendering as expected

- With the add-in sideloaded and running, paste the URLs for the add-in's ribbon icons into a browser's navigation bar and see if the icon files open.
- By default, add-in errors connected to the Office UI are suppressed. You can turn on these error messages with the following steps.
 1. With the add-in removed, open the **File** tab of the Office application.
 2. Select **Options**.
 3. In the **Options** dialog, select **Advanced**.
 4. In the **General** section (the **Developers** section for Outlook), enable **Show add-in user interface errors**.

Sideload the add-in again and see if there are any errors.

Changes to add-in commands including ribbon buttons and menu items do not take effect

Clearing the cache helps ensure the latest version of your add-in's manifest is being used. To clear the Office cache, follow the instructions in [Clear the Office cache](#). If you're using Office on the web, clear your browser's cache through the browser's UI.

Add-in commands from old development add-ins stay on ribbon even after the cache is cleared

Sometimes buttons or menus from an add-in that you were developing in the past appears on the ribbon when you run an Office application even after you have cleared the cache. Try these techniques:

- If you develop add-ins on more than one computer and your user settings are synchronized across the computers, try [clearing the Office cache](#) on all the computers. Shut down all Office applications on all the computers, and then clear the cache on all of them before you open any Office application on any of them.
- If you [published the manifest of the old add-in to a network share](#), shut down all Office applications, clear the cache, and then *be sure that the manifest for the add-in is removed from the shared folder*.

Changes to static files, such as JavaScript, HTML, and CSS do not take effect

The browser may be caching these files. To prevent this, turn off client-side caching when developing. The details will depend on what kind of server you are using. In most cases, it involves adding certain headers to the HTTP Responses. We suggest the following set.

- Cache-Control: "private, no-cache, no-store"
- Pragma: "no-cache"
- Expires: "-1"

For an example of doing this in an Node.JS Express server, see [this app.js file](#) [↗](#). For an example in an ASP.NET project, see [this cshtml file](#) [↗](#).

If your add-in is hosted in Internet Information Server (IIS), you could also add the following to the web.config.

```
<system.webServer>
  <staticContent>
    <clientCache cacheControlMode="UseMaxAge"
cacheControlMaxAge="0.00:00:00" cacheControlCustom="must-revalidate" />
  </staticContent>
```

If these steps don't seem to work at first, you may need to clear the browser's cache. Do this through the UI of the browser. Sometimes the Edge cache isn't successfully cleared when you try to clear it in the Edge UI. If that happens, run the following command in a Windows Command Prompt.

Bash

```
del /s /f /q
%LOCALAPPDATA%\Packages\Microsoft.Win32WebViewHost_cw5n1h2txyewy\AC\#!123\IN
etCache\
```

Changes made to property values don't happen and there is no error message

Check the reference documentation for the property to see if it is read-only. Also, the [TypeScript definitions](#) for Office JS specify which object properties are read-only. If you attempt to set a read-only property, the write operation will fail silently, with no error thrown. The following example erroneously attempts to set the read-only property `Chart.id`. See also [Some properties cannot be set directly](#).

JavaScript

```
// This will do nothing, since `id` is a read-only property.
myChart.id = "5";
```

Getting error: "This add-in is no longer available"

The following are some of the causes of this error. If you discover additional causes, please tell us with the feedback tool at the bottom of the page.

- If you're using Visual Studio, there may be a problem with the sideloading. Close all instances of the Office host and Visual Studio. Restart Visual Studio and try pressing `F5` again.

- The add-in's manifest has been removed from its deployment location, such as Centralized Deployment, a SharePoint catalog, or a network share.
- If the add-in only manifest is being used, one of the following may apply.
 - The value of the `ID` element in the manifest has been changed directly in the deployed copy. If for any reason, you want to change this ID, first remove the add-in from the Office host, then replace the original manifest with the changed manifest. You may need to clear the Office cache to remove all traces of the original. See the [Clear the Office cache](#) article for instructions on clearing the cache for your operating system.
 - The add-in's manifest has a `resid` that isn't defined anywhere in the [Resources](#) section of the manifest, or there is a mismatch in the spelling of the `resid` between where it is used and where it is defined in the `<Resources>` section.
 - There is a `resid` attribute somewhere in the manifest with more than 32 characters. A `resid` attribute, and the `id` attribute of the corresponding resource in the `<Resources>` section, cannot be more than 32 characters.
- The add-in has a custom Add-in Command but you are trying to run it on a platform that doesn't support them. For more information, see [Add-in commands requirement sets](#).

Add-in doesn't work on Edge but it works on other browsers

See [Troubleshoot EdgeHTML and WebView2 \(Microsoft Edge\) issues](#).

Excel add-in throws errors, but not consistently

See [Troubleshoot Excel add-ins](#) for possible causes.

Word add-in throws errors or displays broken behavior

See [Troubleshoot Word add-ins](#) for possible causes.

Add-in only manifest schema validation errors in Visual Studio projects

If you're using newer features that require changes to the add-in only manifest file, you may get validation errors in Visual Studio. For example, when adding the **<Runtimes>** element to implement the [shared runtime](#), you may see the following validation error.

The element 'Host' in namespace

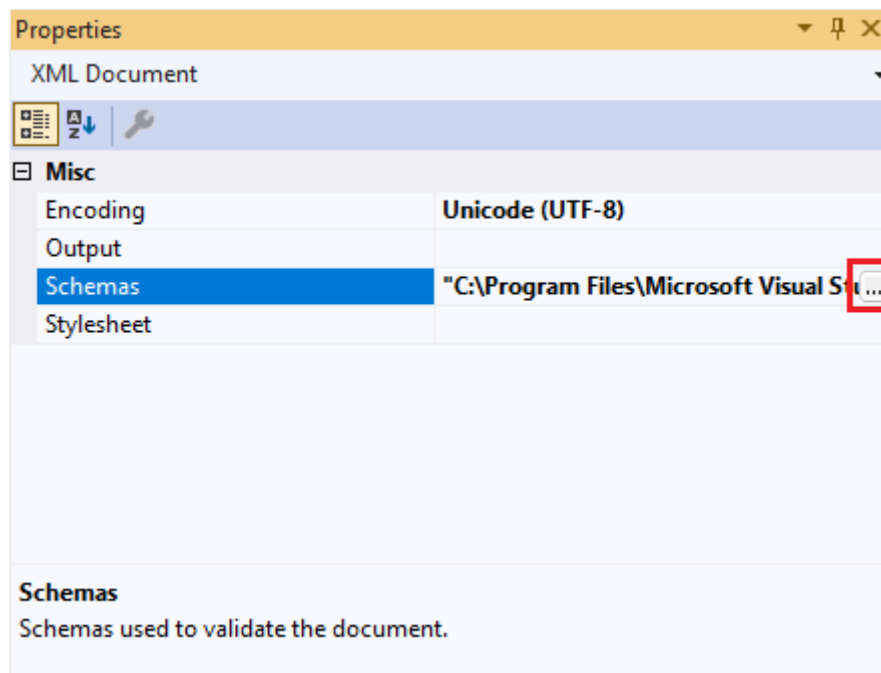
'http://schemas.microsoft.com/office/taskpaneappversionoverrides' has invalid child element 'Runtimes' in namespace

'http://schemas.microsoft.com/office/taskpaneappversionoverrides'

If this occurs, you can update the XSD files that Visual Studio uses to the latest versions. The latest schema versions are at [\[MS-OWEMXML\]: Appendix A: Full XML Schema](#).

Locate the XSD files

1. Open your project in Visual Studio.
2. In **Solution Explorer**, open the manifest.xml file. The manifest is typically in the first project under your solution.
3. Select **View > Properties Window** (**F4**).
4. Set the cursor selection in the manifest.xml so that the **Properties** window shows the **XML Document** properties.
5. In the **Properties** window, select the **Schemas** property, then select the ellipsis (...) to open the **XML Schemas** editor. Here you can find the exact folder location of all schema files your project uses.



Update the XSD files

1. Open the XSD file you want to update in a text editor. The schema name from the validation error will correlate to the XSD file name. For example, open **TaskPaneAppVersionOverridesV1_0.xsd**.
2. Locate the updated schema at [\[MS-OWEMXML\]: Appendix A: Full XML Schema](#). For example, TaskPaneAppVersionOverridesV1_0 is at [taskpaneappversionoverrides Schema](#).
3. Copy the text into your text editor.
4. Save the updated XSD file.
5. Restart Visual Studio to pick up the new XSD file changes.

You can repeat the previous process for any additional schemas that are out-of-date.

When working offline, no Office APIs work

When you're loading the Office JavaScript Library from a local copy instead of from the CDN, the APIs may stop working if the library isn't up-to-date. If you have been away from a project for a while, reinstall the library to get the latest version. The process varies according to your IDE. Choose one of the following options based on your environment.

- **Visual Studio:** Follow these steps to update the NuGet package.
 1. Choose **Tools > NuGet Package Manager > Manage NuGet Packages for Solution**.
 2. Choose the **Updates** tab.
 3. Select "Microsoft.Office.js". Ensure the package source is from nuget.org.
 4. In the left pane, choose **Install** and complete the package update process.
- **Any other IDE:** Get the latest npm packages [@microsoft/office-js](#) and [@types/office-js](#).

See also

- [Debug add-ins in Office on the web](#)
- [Sideload an Office Add-in on Mac](#)
- [Sideload an Office Add-in on iPad](#)
- [Debug Office Add-ins on a Mac](#)
- [Validate an Office Add-in's manifest](#)
- [Debug your add-in with runtime logging](#)
- [Troubleshoot user errors with Office Add-ins](#)
- [Runtimes in Office Add-ins](#)

- [Microsoft Q&A \(Office Development\)](#) 

Debug event-based or spam-reporting add-ins

07/16/2025

This article discusses the key debugging stages to enable and set breakpoints in your code as you implement [event-based activation](#) or [integrated spam reporting](#) in your add-in. Before you proceed, we recommend reviewing the [troubleshooting guide](#) for additional steps on how to resolve development errors.

To begin debugging, select the tab for your applicable client.

Windows (classic)

If you used the [Yeoman generator for Office Add-ins](#) to create your add-in project (for example, by completing an [event-based activation walkthrough](#)), follow the **Created with Yeoman generator** option throughout this article. Otherwise, follow the **Other** steps.

Mark your add-in for debugging and set the debugger port

1. Get your add-in's ID from the manifest.

- **Add-in only manifest:** Use the value of the `<Id>` element child of the root `<OfficeApp>` element.
- **Unified manifest for Microsoft 365:** Use the value of the `"id"` property of the root anonymous `{ ... }` object.

2. In the registry, mark your add-in for debugging.

- **Created with Yeoman generator:** In a command line window, navigate to the root of your add-in folder then run the following command.

```
command line
```

```
npm start
```

In addition to building the code and starting the local server, this command sets the data of the

```
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\Developer\[Add-in
```

[Add-in ID]\UseDirectDebugger registry DWORD value for this add-in to 1. [Add-in ID] is your add-in's ID from the manifest.

- **Other:** In the

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\Developer\[Add-in ID]\UseDirectDebugger registry DWORD value, where [Add-in ID] is your add-in's ID from the manifest, set its data to 1.

❗ **Note**

If the Developer key (folder) doesn't already exist under HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\, complete the following steps to create it.

- a. Right-click (or select and hold) the **WEF** key (folder) and select **New > Key**.
- b. Name the new key **Developer**.

3. In the registry key

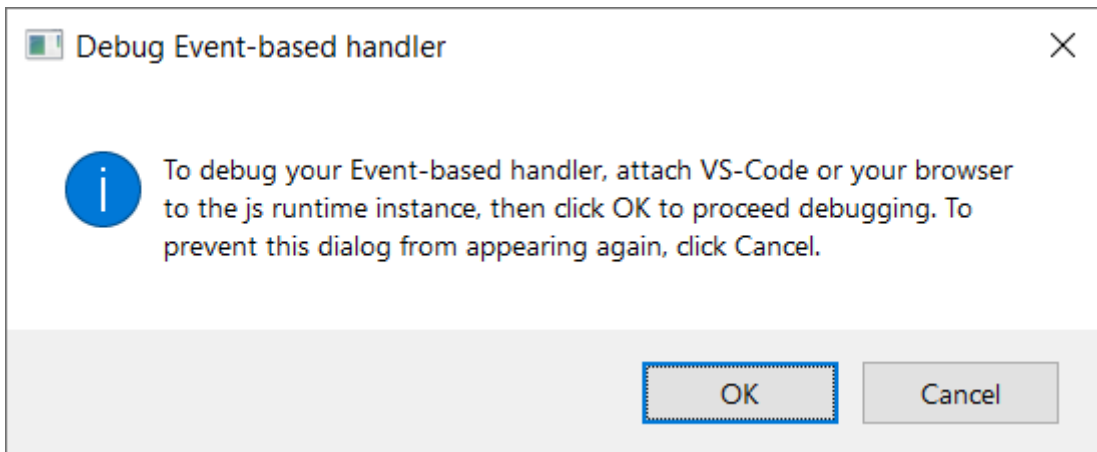
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\Developer\[Add-in ID], where [Add-in ID] is your add-in's ID from the manifest, create a new **DWORD** value with the following configuration.

- **Value name:** DebuggerPort
- **Value data (hexadecimal):** 00002407

This sets the debugger port to 9223.

4. Start your Office application or restart it if it's already open.

5. Perform the action to initiate the event you're developing for, such as creating a new message to initiate the OnNewMessageCompose event or reporting spam messages. The **Debug Event-based handler** dialog should appear. Do *not* interact with the dialog yet.



Configure and attach the debugger

You can debug your add-in using the Microsoft Edge Inspect tool or Visual Studio Code.

Debug with Microsoft Edge

1. Open Microsoft Edge and go to `edge://inspect/#devices`.
2. In the **Remote Target** section, look for your add-in using its ID from the manifest. Then, select **Inspect**.

The DevTools window appears.

ⓘ Note

It may take some time for your add-in to appear in the **Remote Target** section. You may need to refresh the page for the add-in to appear.

3. In the **Sources** tab, go to `file:// > Users/[User]/AppData/Local/Microsoft/Office/16.0/Wef/{[Office profile GUID]}/[Office account encoding]/Javascript/[Add-in ID]_[Add-in Version]_[locale] > bundle.js`. For readability, this article refers to the file name as **bundle.js**, but exact name depends on the Office application.

- Excel: **bundle_excel.js**
- Outlook: **bundle.js**
- PowerPoint: **bundle_powerpoint.js**
- Word: **bundle_word.js**

💡 Tip

There's no direct method to determine the Office profile GUID or mail account encoding used in the **bundle.js** file path. If you're debugging multiple add-ins simultaneously, the easiest way to access an add-in's **bundle.js** file from the DevTools window is to locate the add-in's ID in the file path.

4. In the **bundle.js** file, place breakpoints where you want the debugger to stop.
5. [Run the debugger](#).

Debug with Visual Studio Code

To debug your add-in in Visual Studio Code, you must have at least version 1.56.1 installed.

Configure the debugger

Configure the debugger in Visual Studio Code. Follow the steps applicable to your add-in project.

Created with Yeoman generator

1. In the command line, run the following to open your add-in project in Visual Studio Code.

```
command line
```

```
code .
```

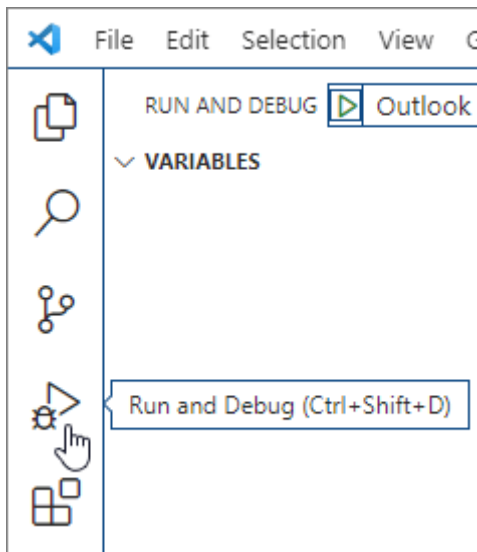
2. In Visual Studio Code, open the `./.vscode/launch.json` file and add the following excerpt to your list of configurations. Save your changes.

```
JSON
```

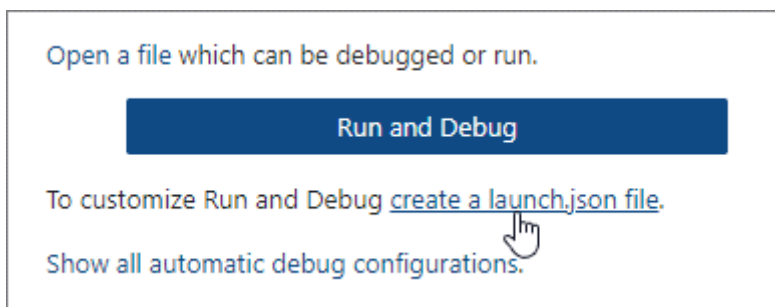
```
{
  "name": "Direct Debugging",
  "type": "node",
  "request": "attach",
  "port": 9223,
  "timeout": 600000,
  "trace": true
}
```

Other

1. Create a new folder called **Debugging** (perhaps in your **Desktop** folder).
2. Open Visual Studio Code.
3. Go to **File > Open Folder**, navigate to the folder you created, then choose **Select Folder**.
4. On the Activity Bar, select **Run and Debug** (**Ctrl** + **Shift** + **D**).



5. Select the **create a launch.json file** link.



6. In the **Select Environment** dropdown, select **Edge: Launch** to create a launch.json file.
7. Add the following excerpt to your list of configurations. Save your changes.

JSON

```
{
  "name": "Direct Debugging",
  "type": "node",
  "request": "attach",
  "port": 9223,
  "timeout": 60000,
```

```
"trace": true
}
```

Attach the debugger

The **bundle.js** file of an add-in contains the JavaScript code of your add-in. It's created when an Office on Windows application is opened. When Office starts, the **bundle.js** file of each installed add-in is cached in the **Wef** folder of your machine.

1. To find the add-in's **bundle.js** file, navigate to the following folder in File Explorer. The text enclosed in `[]` represents your applicable Office and add-in information.

text

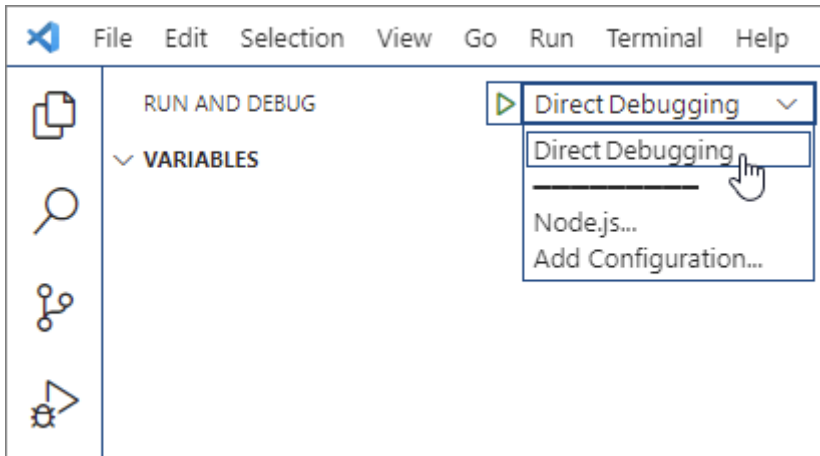
```
%LOCALAPPDATA%\Microsoft\Office\16.0\Wef\{[Office profile GUID]}\[Office  
account encoding]\Javascript\[Add-in ID]\[Add-in Version]\[locale]
```

Tip

- For readability, this article refers to the file name as **bundle.js**, but exact name depends on the Office application.
 - Excel: **bundle_excel.js**
 - Outlook: **bundle.js**.
 - PowerPoint: **bundle_powerpoint.js**
 - Word: **bundle_word.js**
- There's no direct method to determine the Office profile GUID and account encoding used in the **bundle.js** file path. The most effective approach to locate your add-in's **bundle.js** file is to manually inspect each folder until you locate the **Javascript** folder that contains your add-in's ID.
- The **bundle.js** file is downloaded to the local **Wef** folder when the add-in is first installed. It's refreshed every time the Office application starts or is restarted. If the **bundle.js** file doesn't appear in the **Wef** folder and your add-in is installed or sideloaded, restart Office. For Outlook, you may need to [remove your add-in](#), then [sideload](#) it again.

2. Open **bundle.js** in Visual Studio Code.
3. Place breakpoints in **bundle.js** where you want the debugger to stop.

4. In the **DEBUG** dropdown, select **Direct Debugging**, then select the **Start Debugging** icon.



Run the debugger

After confirming that the debugger is attached, return to the Office application. In the **Debug Event-based handler** dialog, select **OK**.

You can now reach your breakpoints to debug your event-based activation or spam-reporting code.

Important

Starting in Version 2403 (Build 17425.20000), event-based and spam-reporting add-ins use the [V8 JavaScript engine](#) to run JavaScript, regardless of whether debugging is turned on or off. In earlier versions, the Chakra JavaScript engine is used when debugging is off, but the V8 engine may be used when debugging is turned on.

Stop the debugger

To stop debugging the rest of the current Office on Windows session, in the **Debug Event-based handler** dialog, choose **Cancel**. To re-enable debugging, restart the Office application.

To prevent the **Debug Event-based handler** dialog from popping up and stop debugging for subsequent sessions, delete the associated registry key,

`HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\WEF\Developer\[Add-in ID]\UseDirectDebugger`, or set its value to `0`.

Stop the local server

When you want to stop the local web server and uninstall the add-in, follow the applicable instructions:

- To stop the server, run the following command. If you used `npm start`, the following command should also uninstall the add-in.

command line

```
npm stop
```

- If you manually sideloaded the add-in, see [Remove a sideloaded add-in](#).

See also

- [Activate add-ins with events](#)
- [Implement an integrated spam-reporting add-in](#)
- [Troubleshoot event-based and spam-reporting add-ins](#)
- [Debug your add-in with runtime logging](#)