

PowerToys Command Palette utility

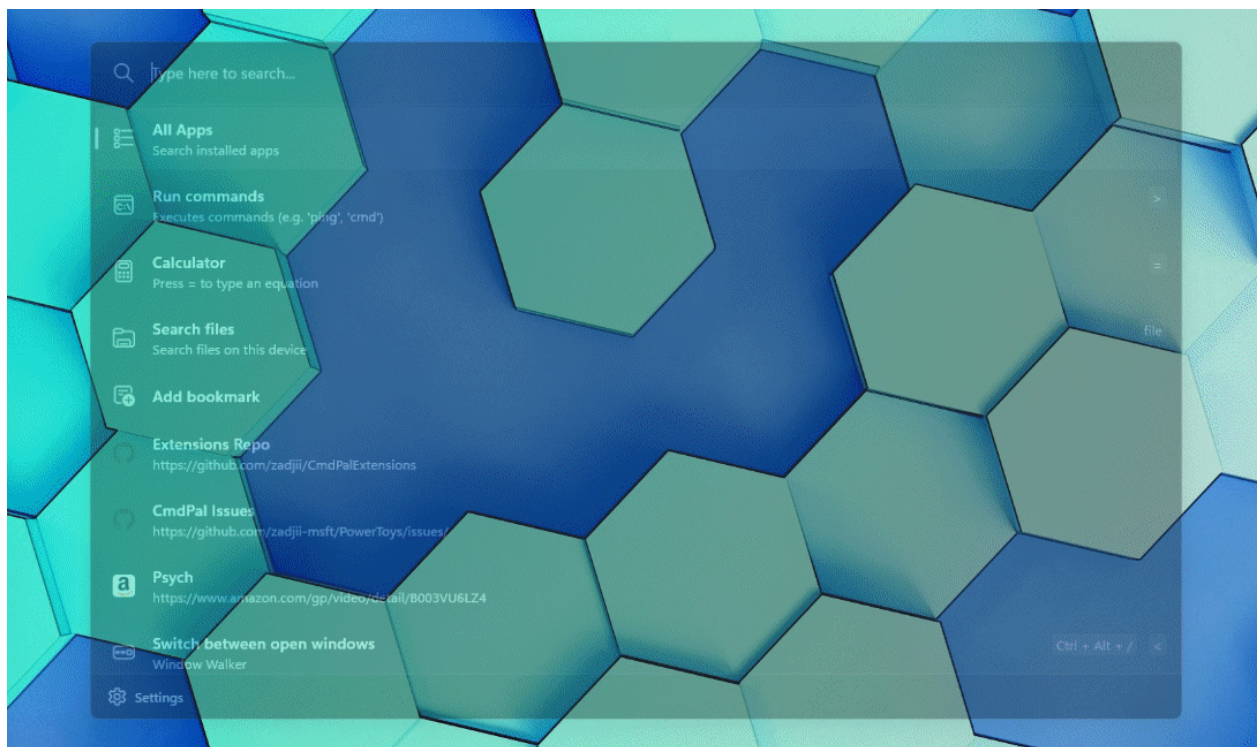
Article • 03/31/2025

PowerToys Command Palette allows you to easily access all of your most frequently used commands, apps, and development tools - all from a single solution that is fast, customizable to your unique preferences, and extensible to include your favorite apps. The Command Palette is intended to be the successor of [PowerToys Run](#).

To use the Command Palette, select `Win + Alt + Space` and start typing! (Note that the keyboard shortcut can be changed in the settings window.)

Important

For this utility to work, the Command Palette must be enabled and running in the background.



Features

Command Palette features include:

- Search for applications, folders or files
- Run commands using `>` (for example, `> cmd` will launch Command prompt, or `> Shell:startup` will open the Windows startup folder)
- Switch between open windows (previously known as [Window Walker](#))

- Do a simple calculation using calculator
- Add bookmarks for frequently visited webpages
- Execute system commands
- Open web pages or start a web search
- Rich extensions to add additional commands and features easily

Settings

The following general options are available within the Command Palette settings page. You can open the settings page by using the Command Palette.

 Expand table

Setting	Description
Activation shortcut	Define the keyboard shortcut to show/hide the Command Palette.
Go home when activated	When the Command Palette is activated it will return to the home page.
Highlight search on activate	The previous search text will be selected when the Command Palette is opened.
Show app details	App details are automatically expanded when displaying an app as a result.
Backspace goes back	Typing <code>Backspace</code> will take you back to the previous page.
Single-click activates	Activate list items with a single click. When disabled, single clicking selects the item and double clicking activates it.

Related content

- [PowerToys Run](#)
- [Extensibility overview](#)
- [Extension samples](#)

Extensibility overview

Article • 03/31/2025

The Command Palette provides a full extension model, allowing developers to create their own experiences for the palette.

The fastest way to get started writing extensions is from the Command Palette itself. Just run the "Create a new extension" command, fill out the fields to populate the template project, and you should be ready to start.

For more detailed instructions, you can follow these pages:

- [Creating an extension](#)
- [Adding commands](#)
- [Update a list of commands](#)
- [Add top-level commands to your extension](#)
- [Command results](#)
- [Display markdown content](#)
- [Get user input with forms](#)

Extension details

Command Palette defines a WinRT API ([Microsoft.CommandPalette.Extensions](#)), which is how extensions can communicate with Command Palette.

Command Palette will use the Package Catalog to find apps that list themselves as an `windows.appExtension` for `com.microsoft.commandpalette`.

Registering your extension

Extensions can register themselves with the Command Palette using their `.appxmanifest`. As an example:

XML

```
<Extensions>
  <com:Extension Category="windows.comServer">
    <com:ComServer>
      <com:ExeServer Executable="ExtensionName.exe" Arguments="-
RegisterProcessAsComServer" DisplayName="Sample Extension">
        <com:Class Id="<Extension CLSID Here>" DisplayName="Sample
Extension" />
      </com:ExeServer>
    </com:ComServer>
  </com:Extension>
</Extensions>
```

```

        </com:ComServer>
    </com:Extension>
    <uap3:Extension Category="windows.appExtension">
        <uap3:AppExtension Name="com.microsoft.commandpalette"
            Id="YourApplicationUniqueId"
            PublicFolder="Public"
            DisplayName="Sample Extension"
            Description="Sample Extension for Command
Palette">
            <uap3:Properties>
                <CmdPalProvider>
                    <Activation>
                        <CreateInstance ClassId="<Extension CLSID Here>" />
                    </Activation>
                    <SupportedInterfaces>
                        <Commands />
                    </SupportedInterfaces>
                </CmdPalProvider>
            </uap3:Properties>
        </uap3:AppExtension>
    </uap3:Extension>
</Extensions>

```

In this manifest, we're using an out-of-process COM server to act as the communication layer between your app and Command Palette. **Don't worry about this!** The template project will take care of creating a COM server for you, starting it, and marshalling your objects to Command Palette.

Important notes

Some notable elements about the manifest example:

- The application must specify a `Extensions.uap3Extension.AppExtension` with the Name set to `com.microsoft.commandpalette`. This is the unique identifier which Command Palette uses to find it's extensions.
- The application must specify a `Extensions.comExtension.ComServer` to host their COM class. This allows for the OS to register that GUID as a COM class we can instantiate.
 - Make sure that this CLSID is unique, and matches the one in your application. If you change one, you need to change all three.
- In the `Properties` of your `AppExtension`, you must specify a `CmdPalProvider` element. This is where you specify the CLSID of the COM class that Command Palette will instantiate to interact with your extension.
 - Currently, only `Commands` is supported.

Related content

- [PowerToys Command Palette utility](#)
- [Extension samples](#)

Creating an extension

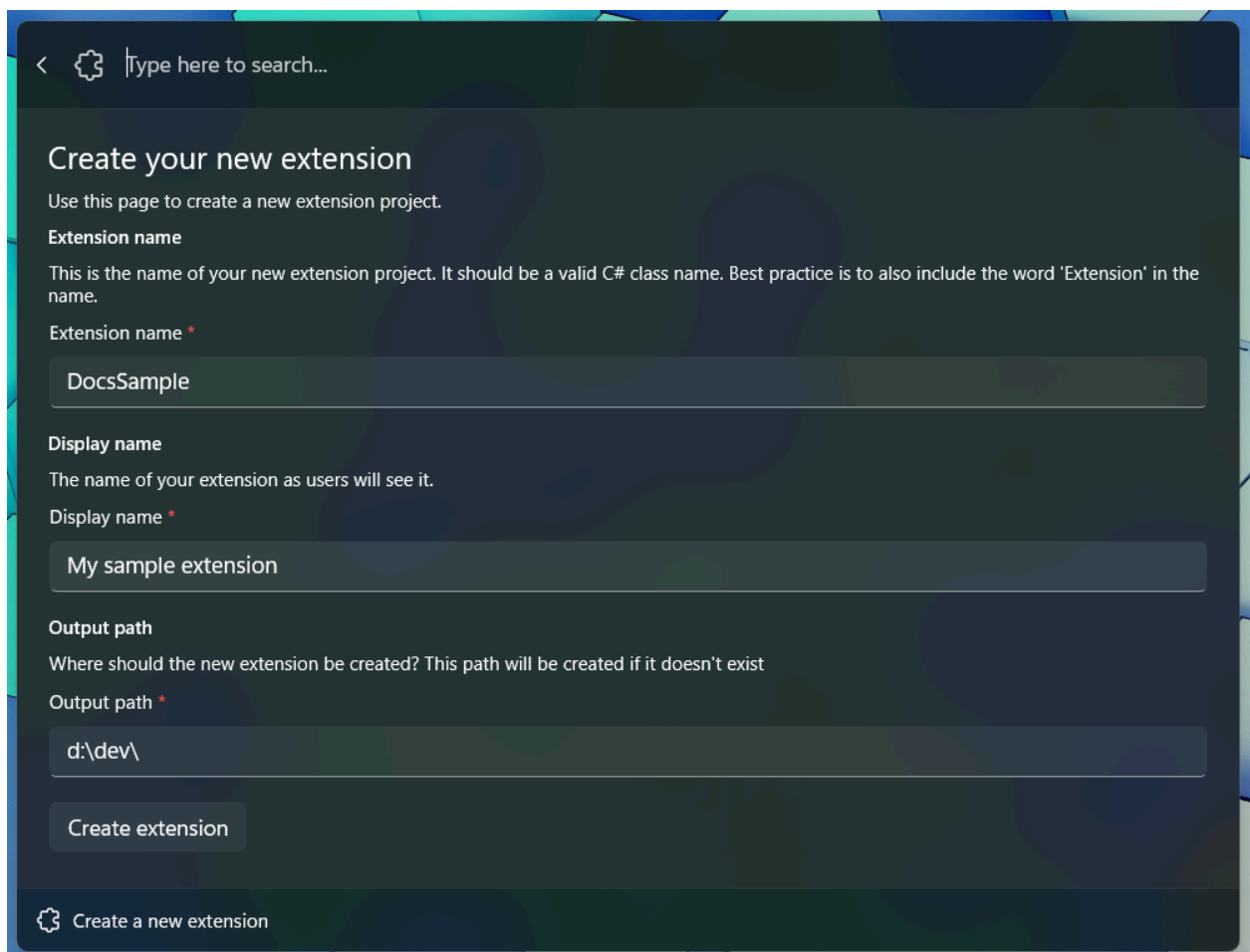
Article • 03/31/2025

The fastest way to get started writing extensions is from the Command Palette itself. Just run the "Create a new extension" command, fill out the fields to populate the template project, and you should be ready to start.

Create a new extension

The form will ask you for the following information:

- **ExtensionName:** The name of your extension. This will be used as the name of the project and the name of the class that implements your commands. Make sure it's a valid C# class name - it shouldn't have any spaces or special characters, and should start with a capital letter.
- **Extension Display Name:** The name of your extension as it will appear in the Command Palette. This can be a more human-readable name.
- **Output Path:** The folder where the project will be created.
 - The project will be created in a subdirectory of the path you provided.
 - If this path doesn't exist, it will be created for you.



The screenshot shows the 'Create your new extension' dialog box in Visual Studio. At the top, there is a search bar with a magnifying glass icon and the text 'Type here to search...'. Below this, the title 'Create your new extension' is displayed, followed by the instruction 'Use this page to create a new extension project.'.

The form contains three main sections:

- Extension name:** A text input field with the label 'Extension name' and a red asterisk. Below the label is a descriptive sentence: 'This is the name of your new extension project. It should be a valid C# class name. Best practice is to also include the word 'Extension' in the name.' The input field contains the text 'DocsSample'.
- Display name:** A text input field with the label 'Display name' and a red asterisk. Below the label is a descriptive sentence: 'The name of your extension as users will see it.' The input field contains the text 'My sample extension'.
- Output path:** A text input field with the label 'Output path' and a red asterisk. Below the label is a descriptive sentence: 'Where should the new extension be created? This path will be created if it doesn't exist'. The input field contains the text 'd:\dev\'.

At the bottom of the form is a 'Create extension' button. Below the button is a footer bar with a magnifying glass icon and the text 'Create a new extension'.

Once you submit the form, Command Palette will automatically generate the project for you. At this point, your projects structure should look like the following:

plaintext

```
ExtensionName/
|   Directory.Build.props
|   Directory.Packages.props
|   nuget.config
|   ExtensionName.sln
└── ExtensionName
    |   app.manifest
    |   Package.appxmanifest
    |   Program.cs
    |   ExtensionName.cs
    |   ExtensionName.csproj
    |   ExtensionNameCommandsProvider.cs
    ├── Assets
    |   <A bunch of placeholder images>
    ├── Pages
    |   ExtensionNamePage.cs
    ├── Properties
    |   |   launchSettings.json
    |   └── PublishProfiles
    |       |   win-arm64.pubxml
    |       |   win-x64.pubxml
```

(with `ExtensionName` replaced with the name you provided)

From here, you can immediately build the project and run it. Once your package is deployed and running, Command Palette will automatically discover your extension and load it into the palette.

💡 Tip

Make sure you *deploy* your app! Just **building** your application won't update the package in the same way that deploying it will.

⚠ Warning

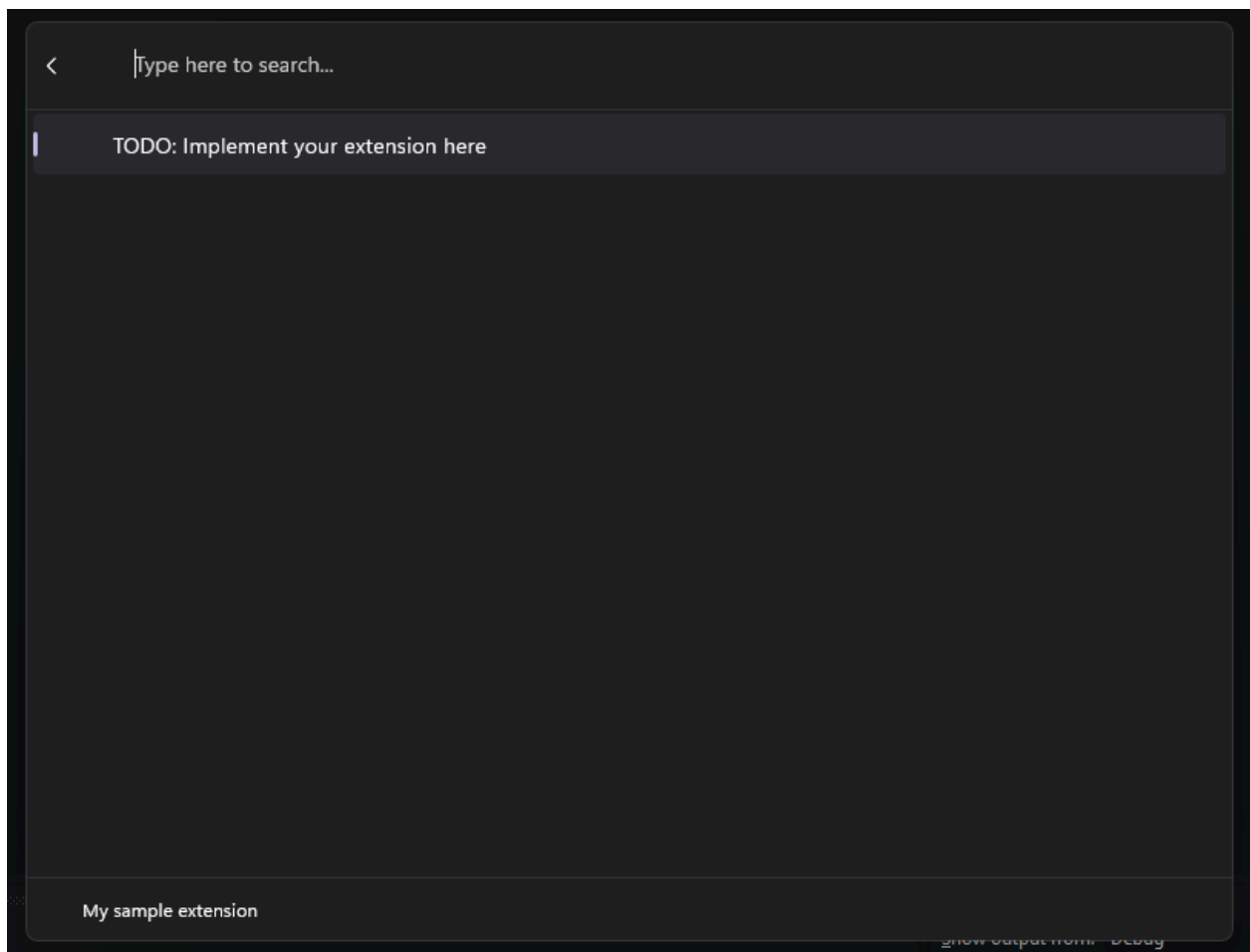
Running "ExtensionName (Unpackaged)" from Visual Studio will not **deploy** your app package.

If you're using `git` for source control, and you used the standard `.gitignore` file for C#, you'll want to remove the following two lines from your `.gitignore` file:

```
**/Properties/launchSettings.json  
*.pubxml
```

These files are used by WinAppSdk to deploy your app as a package. Without it, anyone who clones your repo won't be able to deploy your extension.

You should be able to see your extension in the Command Palette at the end of the list of commands. Entering that command should take you to the page for your command, and you should see a single command that says `TODO: Implement your extension here`.



Congrats! You've made your first extension! Now let's go ahead and actually add some commands to it.

When you make changes to your extension, you can rebuild your project and deploy it again. Command Palette will **not** notice changes to packages that are re-ran through Visual Studio, so you'll need to manually run the "**Reload**" command to force Command Palette to re-instantiate your extension.

Next up: [Add commands to your extension](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Adding commands to your extension

Article • 03/31/2025

Previous: [Creating an extension](#). We'll be starting with the project created in that article.

Now that you've created your extension, it's time to add some commands to it.

Add some commands

We can start by navigating to the `ExtensionNamePage.cs` file. This file is the [ListPage](#) that will be displayed when the user selects your extension. In there you should see:

C#

```
public ExtensionNamePage()
{
    Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
    Title = "My sample extension";
    Name = "Open";
}
public override IListItem[] GetItems()
{
    return [
        new ListItem(new NoOpCommand()) { Title = "TODO: Implement your extension here" }
    ];
}
```

Here you can see that we've set the icon for the page, the title, and the name that's shown at the top-level when you have the command selected. The **GetItems** method is where you'll return the list of commands that you want to show on this page. Right now, that's just returning a single command that does nothing. Let's instead try making that command open *this* page in the user's default web browser.

We can change the implementation of **GetItems** to the following:

C#

```
public override IListItem[] GetItems()
{
    var command = new
    OpenUrlCommand("https://learn.microsoft.com/windows/powertoys/command-
    palette/adding-commands");
    return [
        new ListItem(command)
    ]
}
```

```

        Title = "Open the Command Palette documentation",
    }
];
}

```

Re-deploy your app, run the "reload" command to refresh the extensions in the palette, and head to your extension. You should see that the command now opens the Command Palette documentation.

The **OpenUrlCommand** is a helper for opening a URL in the user's default web browser. You can also implement an extension however you want. Let's instead make a new command, that shows a **MessageBox**. To do that, we need to create a new class that implements **IInvokableCommand**.

C#

```

using System.Runtime.InteropServices;

namespace ExtensionName;

internal sealed partial class ShowMessageCommand : IInvokableCommand
{
    public override string Name => "Show message";
    public override IconInfo Icon => new("\uE8A7");

    public override CommandResult Invoke()
    {
        // 0x00001000 is MB_SYSTEMMODAL, which will display the message box
        // on top of other windows.
        _ = MessageBox(0, "I came from the Command Palette", "What's up?",
            0x00001000);
        return CommandResult.KeepOpen();
    }

    [DllImport("user32.dll", CharSet = CharSet.Unicode)]
    public static extern int MessageBox(IntPtr hWnd, string text, string
        caption, uint type);
}

```

Now we can add this command to the list of commands in the `ExtensionNamePage.cs` file:

C#

```

public override IListItem[] GetItems()
{
    var command = new
        OpenUrlCommand("https://learn.microsoft.com/windows/powertoys/command-

```

```
palette/creating-an-extension");
var showMessageCommand = new ShowMessageCommand();
return [
    new ListItem(command)
    {
        Title = "Open the Command Palette documentation",
    },
    new ListItem(showMessageCommand),
];
}
```

Deploy and reload, and presto - a command to show a message box!

Tip

At about this point, you'll probably want to initialize a git repo / {other source control method of your choice} for your project. This will make it easier to track changes, and to share your extension with others.

We recommend using GitHub, as it's easy to collaborate on your extension with others, and get feedback, and share it with the world.

Adding more pages

So far, we've only worked with commands that "do something". However, you can also add commands that show additional pages within the Command Palette. There are basically two types of "Commands" in the Palette:

- **InvokableCommand** - These are commands that *do something*.
- **IPage** - These are commands that *show something*.

Because **IPage** implementations are **ICommand**'s, you can use them anywhere you can use commands. This means you can add them to the top-level list of commands, or to a list of commands on a page, the context menu on an item, etc.

There are two different kinds of pages you can show:

- **ListPage** - This is a page that shows a list of commands. This is what we've been working with so far.
- **ContentPage** - This is a page that shows rich content to the user. This allows you to specify abstract content, and let Command Palette worry about rendering the content in a native experience. There are two different types of content supported so far:

- [Markdown content](#) - This is content that's written in Markdown, and is rendered in the Command Palette. See [MarkdownContent](#) for details.
- [Form content](#) - This is content that shows a form to the user, and then returns the results of that form to the extension. These are powered by [Adaptive Cards](#) [↗](#). This is useful for getting user input, or displaying more complex layouts of information. See [FormContent](#) for details.

Start by adding a new page that shows a list of commands. Create a new class that implements **ListPage**:

```
C#

using Microsoft.CommandPalette.Extensions.Toolkit;
using System.Linq;

namespace ExtensionName;

internal sealed partial class MySecondPage : ListPage
{
    public MySecondPage()
    {
        Icon = new("\uF147"); // Dial2
        Title = "My second page";
        Name = "Open";
    }

    public override IListItem[] GetItems()
    {
        // Return 100 CopyText commands
        return Enumerable
            .Range(0, 100)
            .Select(i => new ListItem(new CopyTextCommand($"{i}")))
            {
                Title = $"Copy text {i}"
            }.ToArray();
    }
}
```

Next, update the `ExtensionNamePage.cs` to include this new page:

```
diff

    public override IListItem[] GetItems()
    {
        OpenUrlCommand command =
        new("https://learn.microsoft.com/windows/powertoys/command-palette/creating-
        an-extension");
        return [
            new ListItem(command)
        ]
    }
```

```
        Title = "Open the Command Palette documentation",
    },
    new ListItem(new ShowMessageCommand()),
+    new ListItem(new MySecondPage()) { Title = "My second page",
Subtitle = "A second page of commands" },
    ];
}
```

Deploy, reload, and you should now see a new page in your extension that shows 100 commands that copy a number to the clipboard.

Next up: [Update a list of commands](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Update a list of commands

Article • 03/31/2025

Previous: [Adding commands](#).

So far we've shown how to return a list of static items in your extension. However, your items can also change, to show real-time data, or to reflect the state of the system. In this article, we'll show you how to update the list of commands in your extension.

Updating a command

Almost all extension objects in the Command Palette implement the **IPropChanged** interface. This allows them to notify the Command Palette when they've changed, and the Command Palette will update the UI to reflect those changes. If you're using the toolkit implementations, this interface has already been implemented for you for properties that support it.

As a simple example, you can update the title of the page. To do this, you can add a command which will simply update the title of the page.

C#

```
public override IListItem[] GetItems()
{
    OpenUrlCommand command =
    new("https://learn.microsoft.com/windows/powertoys/command-palette/creating-
    an-extension");

    AnonymousCommand updateCommand = new(action: () => { Title += " Hello";
    }) { Result = CommandResult.KeepOpen() };

    return [
        new ListItem(command)
        {
            Title = "Open the Command Palette documentation",
        },
        new ListItem(updateCommand),
    ];
}
```

Here, we're using **AnonymousCommand** to create a command that will update the title of the page. **AnonymousCommand** is a helper that's useful for creating simple, lightweight commands that don't need to be reused.

You can of course create custom **ListItem** objects too:

C#

```
internal sealed partial class IncrementingListItem : ListItem
{
    public IncrementingListItem() :
        base(new NoOpCommand())
    {
        Command = new AnonymousCommand(action: Increment) { Result =
CommandResult.KeepOpen() };
        Title = "Increment";
    }

    private void Increment()
    {
        Subtitle = $"Count = {++_count}";
    }

    private int _count;
}
```

diff

```
    public override IListItem[] GetItems()
    {
        OpenUrlCommand command =
new("https://learn.microsoft.com/windows/powertoys/command-palette/creating-
an-extension");
        return [
            new ListItem(command)
            {
                Title = "Open the Command Palette documentation",
            },
            new ListItem(new ShowMessageCommand()),
+           new IncrementingListItem(),
        ];
    }
```

You're on your way to creating your own idle clicker game, as a Command Palette extension.

Updating the list of commands

You can also change the list of items on the page. This can be useful for pages that load results asynchronously, or for pages that show different commands based on the state of the app.

To do this, you can use the **RaiseItemsChanged** method on the **ListPage** object. This will notify the Command Palette that the list of items has changed, and it should re-fetch

the list of items. As an example, let's make the **IncrementingListItem** from above update the list of items on the page.

Update your list item to take a reference to the page, and add a method to increment the count:

C#

```
internal sealed partial class IncrementingListItem : ListItem
{
    public IncrementingListItem(ExtensionNamePage page) :
        base(new NoOpCommand())
    {
        _page = page;
        Command = new AnonymousCommand(action: _page.Increment) { Result =
        CommandResult.KeepOpen() };
        Title = "Increment";
    }

    private ExtensionNamePage _page;
}
```

Then, change your page as follows:

cs

```
public ExtensionNamePage()
{
    Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
    Title = "My sample extension";
    Name = "Open";

    _items = [new IncrementingListItem2this) { Subtitle = $"Item 0" }];
}
public override IListItem[] GetItems()
{
    return _items.ToArray();
}
internal void Increment()
{
    _items += new IncrementingListItem(this) { Subtitle = $"Item
{_items.Count}" };
    RaiseItemsChanged();
}
private List<ListItem> _items;
```

Now, every time you perform one of the **IncrementingListItem** commands, the list of items on the page will update to add another item. We're using a single **List** owned by the page to own all the items. Notably, we're creating the new items in the **Increment**

method, before calling **RaiseItemsChanged**. The **Invoke** of a **IInvokableCommand** can take as long as you'd like. All your code is running in a separate process from the Command Palette, so you won't block the UI. But constructing the items before calling **RaiseItemsChanged** will help keep your extension *feeling* more responsive.

Showing a loading spinner

Everything so far has been pretty instantaneous. Many extensions however may need to do some work that takes a lot longer. In that case, you can set **Page.IsLoading** to `true` to show a loading spinner. This will help indicate that the extension is doing something in the background.

C#

```
internal void Increment()
{
    Page.IsLoading = true;
    Task.Run(() =>
    {
        Thread.Sleep(5000);
        _items += new IncrementingListItem(this) { Subtitle = $"Item
{_items.Count}" };
        RaiseItemsChanged();
        Page.IsLoading = false;
    });
}
```

Best practice is to set **IsLoading** to `true` before starting the work. Then do all the work to build all the new **ListItems** you need to display to the user. Then, once the items are ready, call **RaiseItemsChanged** and set **IsLoading** back to `false`. This will ensure that the loading spinner is shown for the entire duration of the work, and that the UI is updated as soon as the work is done (without waiting for your extension to construct new **ListItems** objects).

Next up: [Add top-level commands to your extension](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Adding top-level commands to your extension

Article • 03/31/2025

Previous: [Update a list of commands.](#)

So far, you've only added commands to a single page within your extension. You can also add more commands directly to the top-level list of commands too.

Adding the top-level commands

To do that, head on over to the `ExtensionNameCommandsProvider.cs` file. This file is where you'll add commands that should be shown at the top-level of the Command Palette. As you can see, there's currently only a single item there:

C#

```
public ExtensionNameCommandsProvider()
{
    DisplayName = "My sample extension";
    Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
    _commands = [
        new CommandItem(new ExtensionNamePage()) { Title = DisplayName },
    ];
}

public override ICommandItem[] TopLevelCommands()
{
    return _commands;
}
```

This sample extension creates a list of commands when the extension is created and returns that list whenever it's asked for the top-level commands. This prevents the extension from re-creating the list of commands every time the top-level commands are requested. This is a performance optimization.

If you want to add another command to the top-level list of commands, you can add another **CommandItem**:

C#

```
public ExtensionNameCommandsProvider()
{
    DisplayName = "My sample extension";
```

```
Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
_commands = [
    new CommandItem(new ExtensionNamePage()) { Title = DisplayName },
    new CommandItem(new ShowMessageCommand()) { Title = "Send a message"
},
];
}
```

There you have it. Now you can add additional top-level commands to your extension.

If you'd like to update the list of top-level commands dynamically, you can do so in the same way as you would update a list page. This can be useful for cases like an extension that might first require the user to log in, before showing certain commands. In that case, you can show the "log in" command at the top level initially. Then, once the user logs in successfully, you can update the list of top-level commands to include the commands that required authentication.

Once you've determined that you need to change the top level list, call [RaiseItemsChanged](#) on your **CommandProvider**. Command Palette will then request the top-level commands via **TopLevelCommands** again, and you can return the updated list.

Tip

Create the **CommandItem** objects for the top-level commands before calling **RaiseItemsChanged**. This will ensure that the new commands are available when Command Palette requests the top-level commands. This will ensure that the work being executed in each call to **TopLevelCommands** method to a minimum.

Next up: [Command Results](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Command results

Article • 03/31/2025

Previous: [Add top-level commands to your extension](#)

An [IInvokableCommand](#) is a fundamental unit of *do something* in the Command Palette. The [Invoke](#) method is called when the user selects the command, and it's where you *do something* in your extension. The [Invoke](#) method returns an **ICommandResult**, which tells the Command Palette what to do after the command has been invoked. This page details what's possible with each type of command result.

The toolkit provides a number of helper methods to create command results. These are all static methods on the **CommandResult** class. Calling these methods on their own won't do anything. You must return those objects as the result of a [Invoke](#) method, for Command Palette to handle them.

KeepOpen command result

The **KeepOpen** command result does nothing. It leaves the palette in its current state, with the current page stack and query. This can be useful for commands that want to keep the the user in the Command Palette, to keep working with the current page.

ⓘ Note

Even when returning **KeepOpen**, launching a new app or window from the Command Palette will automatically hide the palette the next window receives focus.

Hide command result

This command result keeps the current page open, but hides the Command Palette. This can be useful for commands that want to take the user briefly out of the Command Palette, but then come back to this context.

GoBack command result

This result takes the user back a page in the Command Palette, and keeps the window visible. This is perfect for form pages, where doing the command should take you the user back to the previous context.

GoHome command result

This result takes the user back to the main page of the Command Palette. It will leave the Palette visible (unless the palette otherwise loses focus). Consider using this for scenarios where you've changed your top-level commands.

Dismiss command result

This result hides the Command Palette after the action is executed, and takes it back to the home page. On the next launch, the Command Palette will start from the main page with a blank query. This is useful for commands that are one-off actions, or that don't need to keep the Command Palette open.

If you don't know what else to use, this should be your default. Ideally, users should come into the palette, find what they need, and be done with it.

ShowToast command result

This result displays a transient desktop-level message to the user. This is especially useful for displaying confirmation that an action took place when the palette will be closed.

Consider the [CopyTextCommand](#) in the helpers - this command will show a toast with the text "Copied to clipboard", then dismiss the palette.

By default, [CommandResult.ShowToast\(string\)](#) helper will have a **Result** of `CommandResult.Dismiss`. However, you can instead change the result to any of the other results if you want. This allows you to display a toast and keep the palette open, if you'd like.

Confirm command result

This result displays a confirmation dialog to the user. If the user confirms the dialog, then the **PrimaryCommand** of the *ConfirmationArgs* will be performed.

This is useful for commands that might have destructive actions, or that need to confirm user intent.

Example

As an example, here's a page with one command for each kind of command result:

C#

```
using Microsoft.CommandPalette.Extensions.Toolkit;

internal sealed partial class CommandResultsPage : ListPage
{
    public CommandResultsPage()
    {
        Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
        Title = "Example command results";
        Name = "Open";
    }

    public override IListItem[] GetItems()
    {
        ConfirmationArgs confirmArgs = new()
        {
            PrimaryCommand = new AnonymousCommand(
                () =>
                {
                    ToastStatusMessage t = new("The dialog was confirmed");
                    t.Show();
                })
        {
            Name = "Confirm",
            Result = CommandResult.KeepOpen(),
        },
        Title = "You can set a title for the dialog",
        Description = "Are you really sure you want to do the thing?",
    };

    return
    [
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.KeepOpen() }) { Title = "Keep the palette open" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.Hide() }) { Title = "Hide the palette" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.GoBack() }) { Title = "Go back" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.GoHome() }) { Title = "Go home" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.Dismiss() }) { Title = "Dismiss the palette" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.ShowToast("What's up") }) { Title = "Show a toast" },
        new ListItem(new AnonymousCommand(null) { Result =
        CommandResult.Confirm(confirmArgs) }) { Title = "Confirm something" },
    ];
}
}
```

Next up: [Display markdown content](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Display markdown content

Article • 03/31/2025

Previous: [Command Results](#)

So far, we've only shown how to display a list of commands in a **ListPage**. However, you can also display rich content in your extension, such as markdown. This can be useful for showing documentation, or a preview of a document.

Working with markdown content

[IContentPage](#) (and its toolkit implementation, [ContentPage](#)) is the base for displaying all types of rich content in the Command Palette. To display markdown content, you can use the [MarkdownContent](#) class.

As a simple example, we can create the following page:

C#

```
public class MarkdownPage : ContentPage
{
    public MarkdownPage()
    {
        Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
        Title = "Markdown page";
    }

    public override IContent[] GetContent()
    {
        return [
            new MarkdownContent("# Hello, world!\n This is a **markdown** page."),
        ];
    }
}
```

In this example, a new **MarkdownPage** that displays a simple markdown string is created. The **MarkdownContent** class takes a string of markdown content and renders it in the Command Palette.

You can also add multiple blocks of content to a page. For example, you can add two blocks of markdown content:

C#

```

public override IContent[] GetContent()
{
    return [
        new MarkdownContent("# Hello, world!\n This is a **markdown**
page."),
        new MarkdownContent("## Second block\n This is another block of
content."),
    ];
}

```

This allows you to mix-and-match different types of content on a single page.

Adding commands

You can also add commands to a **ContentPage**. This allows you to add additional commands to be invoked by the user, while in the context of the content. For example, if you had a page that displayed a document, you could add a command to open the document in File Explorer:

C#

```

public class MarkdownExamplePage : ContentPage
{
    public MarkdownExamplePage()
    {
        Icon = new("\uE8A5"); // Document icon
        Title = "Markdown page";
        Name = "Preview file";

        Commands = [
            new CommandContextItem(new
OpenUrlCommand("C:\\Path\\to\\file.txt")) { Title = "Open in File Explorer"
},
        ];
    }
    public override IContent[] GetContent()
    {
        return [
            new MarkdownContent("# Hello, world!\n This is a **markdown**
document.\nI live at `C:\\Path\\to\\file.txt`"),
        ];
    }
}

```

Next up: [Get user input with forms](#)

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Get user input with forms

Article • 03/31/2025

Previous: [Display markdown content](#)

Now that we know how to present basic markdown content, let's try displaying something more elaborate by leveraging the power of [Adaptive Cards](#). This is useful for creating forms, or for displaying more complex content.

Working with forms

Command Palette supports Adaptive Cards, which are a way to create rich, interactive content. This can be useful for creating forms, or for displaying more complex content.

You can create a card in the Command Palette with the **IFormContent** interface (see [FormContent](#) for the toolkit implementation). This allows you to provide the Adaptive Card JSON, and the Command Palette will render it for you. When the user submits the form, Command Palette will call the **SubmitForm** method on your form, with the JSON payload and inputs from the form.

Adaptive card payloads can be created using the [Adaptive Card Designer](#). You can design your card there, and then copy the JSON payload into your extension.

For a full example of using Forms and Content pages, head on over to [SamplePagesExtension/Pages/SampleContentPage.cs](#). Some brief things to note:

- Set the **TemplateJson** property of your form to the JSON payload of your Adaptive Card. (this is the "CARD PAYLOAD EDITOR" value in the Adaptive Card Designer)
- Set the **DataJson** property of your **FormContent** to the data you want to use to fill in your card template. (This is the "SAMPLE DATA EDITOR" value in the Adaptive Card Designer). This is optional, but can make authoring cards easier.
- Implement the **SubmitForm** method to handle the form submission. This method will be called when the user submits the form, and will be passed the JSON payload of the form.

C#

```
public override CommandResult SubmitForm(string payload)
{
    var formInput = JsonNode.Parse(payload)?.AsObject();
    if (formInput == null)
    {
        return CommandResult.GoHome();
    }
}
```

```
}

// retrieve the value of the input field with the id "name"
var name = formInput["name"]?.AsString();

// do something with the data

// and eventually
return CommandResult.GoBack();
}
```

Of course, you can mix and match **IContent** in whatever way you'd like. For example, you could use a block markdown first for the body of a post, and have a **FormContent** next to reply to the post.

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)
- [Extension samples](#)

Publishing your extension

Article • 03/31/2025

The Command Palette provides a full extension model, allowing developers to create their own experiences for the palette. This document provides information about how to publish an extension.

There is a "Sample Project" template included with the Command Palette. This can be used to quickly generate a project that creates a new extension. This will include the `.sln`, `.csproj`, and `.appxmanifest` files needed to create a new extension, as well as the plumbing to get it ready to be published. You will then open the project to the `{ExtensionName}CommandsProvider` class (where `{ExtensionName}` is replaced with the name of your extension project) and implement your commands.

Pre-requisites

The following tools are required to build and publish your extension:

- [Visual Studio 2022](#) (Community, Professional, or Enterprise edition)

WinGet

Publishing packages to WinGet is the recommended way to share your extensions with users. Extension packages which are listed on WinGet can be discovered and installed directly from the Command Palette.

For the most part, following the steps on [Submit packages to Windows Package Manager](#) will get your extension onto WinGet itself.

Before submitting your manifest to WinGet, you'll need to check two things:

Add `windows-commandpalette-extension` tag

Command Palette uses the special `windows-commandpalette-extension` tag to discover extensions. Make sure that your manifest includes this tag, so that Command Palette can discover your extension. Add the following to each `.locale.*.yaml` file in your manifest:

YAML

Tags:

```
- windows-commandpalette-extension
```

Ensure WindowsAppSdk is listed as a dependency

If you're using Windows App SDK, then you'll need to make sure that it is listed as a dependency of your package. Add the following to your `.installer.yaml` manifest:

YAML

```
Dependencies:
  PackageDependencies:
    - PackageIdentifier: Microsoft.WindowsAppRuntime.1.6
```

If you're not using the template project, then this may not apply to you.

Microsoft Store

Command Palette extensions can be published to the Microsoft Store. The process is similar to publishing other apps or extensions. You create a new submission in the Partner Center and upload your `.msix` package. The Command Palette automatically discovers your extension when it's installed from the Microsoft Store.

Command Palette cannot, however, search for & install extensions that are only listed in the store. You can find those by running the following command:

Windows Command Prompt

```
ms-windows-store://assoc/?Tags=AppExtension-com.microsoft.commandpalette
```

You can run this from the "Run commands" command in Command Palette, or from the command-line, or from the Run dialog.

Related content

- [Extensibility overview](#)
- [Extension samples](#)
- [PowerToys Command Palette utility](#)

Extension samples

Article • 03/31/2025

The Command Palette provides a full extension model, allowing developers to create their own experiences for the palette.

For the most up-to-date samples, check out [the samples project on GitHub](#). This contains up-to-date samples of everything that's possible with Command Palette.

Create a command to do something

Create a class that implements `IInvokableCommand` and implement the `Invoke` method. This method will be called when the user selects the command in the Command Palette.

C#

```
class MyCommand :
Microsoft.CommandPalette.Extensions.Toolkit.InvokableCommand {
    public class MyCommand()
    {
        Name = "Do it"; // A short name for the command
        Icon = new("\uE945"); // Segoe UI LightningBolt
    }

    // Open GitHub in the user's default web browser
    public ICommandResult Invoke() {
        Process.Start(new ProcessStartInfo("https://github.com") {
            UseShellExecute = true });

        // Hides the Command Palette window, without changing the page
        // that's open
        return CommandResult.Hide();
    }
}
```

Create a page of commands

The following example shows how to create a page of commands. This page will be shown when the user selects the "Open" command in the Command Palette:

C#

```
using Microsoft.CommandPalette.Extensions.Toolkit;

class MyPage : ListPage {
```



```

public MyPage()
{
    Icon = IconHelpers.FromRelativePath("Assets\\StoreLogo.png");
    Title = "My sample extension";
    Name = "Open";
}

public override IListItem[] GetItems()
{
    return [
        new ListItem(new OpenUrlCommand("https://github.com"))
        {
            Title = "Open GitHub",
        },
        new ListItem(new OpenUrlCommand("https://learn.microsoft.com"))
        {
            Title = "Open Microsoft Learn",
        },
        new ListItem(new
OpenUrlCommand("https://github.com/microsoft/PowerToys"))
        {
            Title = "Open PowerToys on GitHub",
        },
        new ListItem(new CopyTextCommand("Foo bar"))
        {
            Title = "Copy 'Foo bar' to the clipboard",
        },
    ];
}
}

```

Icons

Icons using the [IconInfo](#) class can be specified in a number of ways. Here are some examples:

C#

```

using Microsoft.CommandPalette.Extensions.Toolkit;

namespace ExtensionName;

internal sealed class Icons
{
    // Icons can be specified as a Segoe Fluent icon, ...
    internal static IconInfo OpenFile { get; } = new("\uE8E5"); // OpenFile

    // ... or as an emoji, ...
    internal static IconInfo OpenFileEmoji { get; } = new("📁");
}

```

```
// ... Or as a path to an image file, ...
internal static IconInfo FileExplorer { get; } =
IconHelpers.FromRelativePath("Assets\\FileExplorer.png");

// ... which can be on a remote server, or svg's, or ...
internal static IconInfo FileExplorerSvg { get; } =
new("https://raw.githubusercontent.com/microsoft/PowerToys/refs/heads/main/src/modules/cmdpal/Exts/Microsoft.CmdPal.Ext.Indexer/Assets/FileExplorer.svg");

// Or they can be embedded in a exe / dll:
internal static IconInfo FolderIcon { get; } =
new("%systemroot%\\system32\\system32\\shell32.dll,3");
}
```

Related content

- [PowerToys Command Palette utility](#)
- [Extensibility overview](#)

Command Palette SDK namespaces

Article • 03/31/2025

The Command Palette SDK provides a set of namespaces that contain the classes and interfaces you need to create extensions for the Command Palette. The SDK is designed to be easy to use and provides a consistent experience across all extensions.

SDK namespaces

The following namespaces are available in the Command Palette SDK:

 Expand table

Namespace	Description
Microsoft.CommandPalette.Extensions	Contains the interfaces to create extensions for the Command Palette.
Microsoft.CommandPalette.Extensions.Toolkit	Contains helper classes that make creating extensions easier.

Related content

- [PowerToys Command Palette utility](#)


Microsoft.CommandPalette.Extensions Namespace

Article • 03/31/2025

Contains the interfaces to create extensions for the Command Palette.


These are the raw WinRT interfaces that Command Palette uses to communicate with your extension. These can be implemented however you'd like, in any language that supports implementing WinRT interfaces. For simplicity, there's a reference C# implementation of these interfaces in the [Microsoft.CommandPalette.Extensions.Toolkit](#) namespace.

Structs

 Expand table

Struct	Description
Color	Represents a color value.
KeyChord	Represents a key chord, which is a combination of keys that can be pressed together.
OptionalColor	Represents a color that can be either specified or not specified.

Interfaces

 Expand table

Interface	Description
ICommand	Action a user can take within the Command Palette.
ICommandContextItem	Represents a context menu item for a command.
ICommandItem	Represents an item that can be used in a command.
ICommandProvider	Represents a provider that can be used to create commands.
ICommandResult	Represents the result of a command.
ICommandResultArgs	Represents the arguments for a command result.

Interface	Description
 ICommandSettings	Represents the settings for a command.
 IConfirmationArgs	Represents the arguments for a confirmation dialog.
 IContent	Represents the content of a command.
 IContentPage	Represents a page that can be used in a command.
 IContextItem	Represents a context menu item.
 IDetails	Represents the details of a command.
 IDetailsCommand	Represents a command that contains details.
 IDetailsData	Represents the data that can be used in the details.
 IDetailsElement	Represents an element that can be used in the details.
 IDetailsLink	Represents a link that can be used in the details.
 IDetailsSeparator	Represents a separator that can be used in the details.
 IDetailsTags	Represents the tags that can be used in the details.
 IDynamicListPage	Represents a dynamic list page that can be used in a command.
 IExtension	Represents an extension that can be used in the Command Palette.
 IExtensionHost	Represents the host for an extension.
 IFallbackCommandItem	Represents a fallback command item that can be used in the Command Palette.
 IFallbackHandler	Represents a handler that can be used for fallback commands.
 IFilter	Represents a filter that can be used in the Command Palette.
 IFilters	Represents a collection of filters that can be used in the Command Palette.
 IFilterItem	Represents an item that can be used in a filter.
 IForm	Represents a form that can be used in the Command Palette.
 IFormContent	Represents the content of a form.
 IFormPage	Represents a page that can be used in a form.
 IGoToPageArgs	Represents the arguments for navigating to a page.
 IGridProperties	Represents the properties of a grid.

Interface	Description
IIconData	Represents the data for an icon.
IIconInfo	Represents the information for an icon.
IInvokableCommand	Represents a command that can be invoked.
IItemsChangedEventArgs	Represents the arguments for an items changed event.
IListItem	Represents an item that can be used in a list.
IListPage	Represents a page that can be used in a list.
ILogMessage	Represents a log message.
IMarkdownContent	Represents the content of a markdown page.
IMarkdownPage	Represents a page that can be displayed as markdown.
INotifyItemsChanged	Represents an interface for notifying when items have changed.
INotifyPropertyChanged	Represents an interface for notifying when a property has changed.
IPage	Represents a page that can be used in the Command Palette.
IProgressState	Represents the state of a progress indicator.
IPropertyChangedEventArgs	Represents the arguments for a property changed event.
ISeparatorContextItem	Represents a separator as a context menu item.
ISeparatorFilterItem	Represents a separator as a filter item.
IStatusMessage	Represents a status message.
ITag	Represents a tag that can be used in the Command Palette.
IToastArgs	Represents the arguments for a toast notification.
ITreeContent	Represents the content of a tree.

Enums

[Expand table](#)

Enum	Description
CommandResultKind	Specifies what kind of command it is.
MessageState	Specifies the state of a message.

Enum	Description
NavigationMode	Specifies which navigation direction to take.
ProviderType	Specifies the type of provider. Currently Command is the only type.

Color Struct

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Color** struct represents a color value in the Command Palette. It is used to define colors in the RGB (Red, Green, Blue) color model, with an optional alpha (transparency) channel.

Fields

 Expand table

Field	Type	Description
A	UInt8	Alpha value (transparency)
B	UInt8	Blue value
G	UInt8	Green value
R	UInt8	Red value

CommandResultKind Enum

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **CommandResultKind** enum is used to specify the result of a command execution in the Command Palette. It defines the different actions that can be taken after a command is executed.

Fields

 Expand table

Field	Description
Confirm	Display a confirmation dialog to the user.
Dismiss	Close the Command Palette after the action is executed and dismiss the current state. On the next launch, the Command Palette will start from the main page with a blank query.
GoBack	Navigate to the previous page, and keep it open.
GoHome	Navigate back to the main page of the Command Palette and keep it open. This clears out the current stack of pages, but keeps the palette open.
GoToPage	Navigate to a different page in the palette. The GoToPageArgs will specify which page to navigate to.
Hide	Keep this page open and hide the palette.
KeepOpen	Do nothing. This leaves the palette in its current state, with the current page stack and query.
ShowToast	Display a transient desktop-level message to the user. This is especially useful for displaying confirmation that an action took place when the palette will be closed. Consider the CopyTextCommand in the helpers - this command will show a toast with the text "Copied to clipboard", then dismiss the palette.

ICommand Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Action a user can take within the Command Palette.

Properties

 Expand table

Property	Type	Description
Icon	IIconInfo	Gets the icon of the command.
Id	String	Gets the ID of the command. This is optional but can help support more efficient command lookup in ICommandProvider.GetCommand() .
Name	String	Gets the name of the command.

Example

See the [Add a command](#) page for an example of how to use this interface.

ICommandContextItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ICommandContextItem** interface is used to represent a context menu item in the Command Palette. It is used to define the properties and methods that a context menu item must implement in order to be displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
IsCritical	Boolean	Makes the item red to indicate that it's critical and requires attention.
RequestedShortcut	KeyChord	The shortcut that was requested for this item. This property is used to define the shortcut that will be used to activate the item in the Command Palette.

Remarks

When displaying a **IListItem**'s default **Command** as a context menu item, a new **ICommandContextItem** is created.

ICommandItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ICommandItem** interface is used to represent a command item in the Command Palette. It is used to define the properties and methods that a command item must implement in order to be displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Command	ICommand	The command associated with this item. This is the command that will be executed when the item is activated.
Icon	IIconInfo	The icon associated with this item. This icon will be displayed next to the item in the Command Palette.
MoreCommands	IContextItem[]	An array of additional commands that can be displayed in a submenu. This property is used to define a list of related commands that can be executed from the Command Palette.
Subtitle	String	The subtitle associated with this item. This subtitle will be displayed below the item in the Command Palette.
Title	String	The title associated with this item. This title will be displayed as the main text of the item in the Command Palette.

Remarks

If an **ICommandItem** in a context menu has **MoreCommands**, then activating it will open a submenu with those items. If an **ICommandItem** in a context menu has **MoreCommands** and a non-null **Command**, then activating it will open a submenu with the **Command** first (following the same rules for building a context item from a default **Command**), followed by the items in **MoreCommands**.

When displaying a page:

- The title will be `IPage.Title ?? ICommand.Name`
- The icon will be `ICommand.Icon`

ICommandProvider Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

This is the interface that an extension must implement to provide commands to the Command Palette. The Command Palette will call this interface to get the commands that it should display.

Properties

 [Expand table](#)

Property	Type	Description
DisplayName	String	The display name of the command provider. This is used to identify the provider in the Command Palette.
Frozen	Boolean	Indicates whether the command provider is frozen. A frozen command provider will not be updated or refreshed.
Icon	IconInfo	The icon associated with the command provider. This is used to display an icon in the Command Palette.
Id	String	The unique identifier of the command provider. This is used to identify the provider in the Command Palette.
Settings	ICommandSettings	The settings associated with the command provider. This is used to provide additional information or resources related to the item being displayed in the Command Palette.

Methods

 [Expand table](#)

Method	Description
FallbackCommands()	Returns the fallback commands for the command provider.
GetCommand(String)	Returns the command associated with the specified ID.

Method	Description
InitializeWithHost(IExtensionHost)	Initializes the command provider with the specified host. This is called when the command provider is first created.
TopLevelCommands()	Returns the top-level commands for the command provider.

ICommandProvider.FallbackCommands() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **FallbackCommands** are special top-level items which allow extensions to have dynamic top-level items which respond to the text the user types on the main list page. These are not shown in the top-level list of commands, but are shown when the user types text in the Command Palette. This allows extensions to provide dynamic commands that are not shown in the top-level list.

Returns

An [IFallbackCommandItem\[\]](#) that contains the commands that should be shown in the Command Palette. The commands will be displayed in the order that they are returned by this method.

ICommandProvider.GetCommand(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **GetCommand** method is used to retrieve a command by its ID. This method is used to get a command that has been registered with the Command Palette.

Parameters

id String

The ID of the command to retrieve. The ID is a unique identifier for the command and is used to identify the command in the Command Palette.

Returns

An [ICommand](#) that contains the command that was registered with the Command Palette. If the command is not found, this method returns null.

ICommandProvider.InitializeWithHost(IExtensionHost) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **InitializeWithHost** method is called when the extension is loaded. This method is used to initialize the extension with the host application. The host application provides the extension with a reference to the [IExtensionHost](#) interface, which allows the extension to interact with the host application.

Parameters

host [IExtensionHost](#)

The host application that is loading the extension. The host application provides the extension with a reference to the [IExtensionHost](#) interface, which allows the extension to interact with the host application.

ICommandProvider.TopLevelCommands() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

TopLevelCommands is the method that Command Palette will call to get the list of actions that should be shown when the user opens it. These are the commands that will allow the user to interact with the rest of your extension. They can be simple actions, or they can be pages that the user can navigate to.

Returns

An [ICommandItem\[\]](#) array that contains the commands that should be shown in the Command Palette. The commands will be displayed in the order that they are returned by this method.

ICommandResult Interface

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Indicates what the Command Palette should do after the command is executed. This allows for commands to control the flow of the Command Palette.

Derived [CommandResult](#)

Properties

 Expand table

Property	Type	Description
Args	ICommandResultArgs	The arguments for the command result.
Kind	CommandResultKind	The kind of command result.

ICommandResultArgs Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ICommandResultArgs** interface is used to define the arguments for a command result in the Command Palette.

Properties

ICommandSettings Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ICommandSettings** interface is used to define the settings for a command in the Command Palette. This interface is used to provide additional information or resources related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
SettingsPage	IContentPage	The settings page associated with the command.

IConfirmationArgs Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Derived [ConfirmationArgs](#)

The **IConfirmationArgs** interface is used to define the arguments for a confirmation dialog in the Command Palette. This interface is used to provide additional information or resources related to the item being displayed in the Command Palette.

Properties

 [Expand table](#)

Property	Type	Description
Description	String	The description of the confirmation dialog.
IsPrimaryCommandCritical	Boolean	Indicates whether the primary command is critical.
PrimaryCommand	ICommand	The primary command associated with the confirmation dialog.
Title	String	The title of the confirmation dialog.

IContent Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IContent** interface is used to define a content item in the Command Palette. Content items can be used to display additional information or resources related to the item being displayed in the Command Palette.

Properties

IContentPage Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)


The **IContentPage** interface is used to define a content page in the Command Palette. Content pages can be used to display additional information or resources related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Commands	IContextItem[]	The commands associated with the content page.
Details	IDetails	The details associated with the content page.

Methods

 Expand table

Method	Description
GetContent()	Retrieves the content associated with the content page.

IContentPage.GetContent() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **GetContent** method retrieves the content associated with the content page. This method is used to display additional information or resources related to the item being displayed in the Command Palette.

Returns

An [IContent\[\]](#) array that contains the content associated with the content page.

IContextItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IContextItem** interface is used to define a context menu item in the Command Palette.

Properties

IDetails Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetails** interface is used to define the details view of a command in the Command Palette. Details can be used to provide additional information or resources related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Body	String	The body of the details.
HeroImage	IIconInfo	The hero image associated with the details.
Metadata	IDetailsElement[]	The metadata associated with the details.
Title	String	The title of the details section.

IDetailsCommand Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsCommand** interface is used to define a command in the Command Palette. Commands are used to perform actions or operations related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Command	ICommand	The command associated with the item.

IDetailsData Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsData** interface is used to define data in the Command Palette. Data can be used to provide additional information or resources related to the item being displayed in the Command Palette.

Properties

IDetailsElement Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsElement** interface is used to define an element in the Command Palette. Elements can be used to display additional information or resources related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Data	IDetailsData	The data associated with the element.
Key	String	The key of the element.

IDetailsLink Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsLink** interface is used to define a link in a details page in the Command Palette. Links are used to provide additional information or resources related to the item being displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Link	Windows.Foundation.Uri	The URI of the link.
Text	String	The text to display for the link.

IDetailsSeparator Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsSeparator** interface is used to define a separator in a details page in the Command Palette. Separators are used to visually group items and provide a clear distinction between different sections of the details page.

Properties

IDetailsTags Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDetailsTags** interface is used to define a set of tags on a details page that can be associated with an item in the Command Palette. Tags are used to categorize and organize items, making it easier for users to find and filter them.

Properties

 Expand table

Property	Type	Description
Tags	ITag[]	The tags associated with the item.

IDynamicListPage Interface

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IDynamicListPage** interface is used to define a dynamic list page in the Command Palette. This interface allows extensions to create and manage dynamic lists of items that can be displayed in the Command Palette.

A dynamic list leaves the extension in charge of filtering the list of items.

Properties

 Expand table

Property	Type	Description
SearchText	String	The search text used to filter the list of items. This property is used to define the text that is used to filter the items in the dynamic list.

IExtension Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IExtension** interface is used to define the properties and methods that an extension must implement in order to be used in the Command Palette.

Methods

 Expand table

Method	Description
Dispose()	Disposes of the extension. This method is used to clean up resources and perform any necessary cleanup when the extension is no longer needed.
GetProvider(ProviderType)	Gets the provider type.

IExtension.Dispose() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Dispose** method is used to clean up resources and perform any necessary cleanup when the extension is no longer needed. This method is typically called when the extension is being unloaded or disposed of, allowing it to release any resources it has acquired during its lifetime.

IExtension.GetProvider(ProviderType) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **GetProvider** method is used to retrieve a provider of a specified type. This method allows extensions to access specific providers that are registered with the Command Palette, enabling them to interact with various functionalities and services.

Parameters

providerType [ProviderType](#)

The type of provider to be retrieved. This parameter is used to specify the type of provider that the extension wants to access.

Returns

An **IInspectable** object that represents the provider of the specified type. This object can be used to interact with the provider and access its functionality.

IExtensionHost Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IExtensionHost** interface is used to provide a host for extensions in the Command Palette. It is responsible for managing the lifecycle of extensions, including loading and unloading them, as well as providing access to shared resources and services.

This is an object which extensions shouldn't implement themselves. Rather, this is implemented by the host app itself.

Properties

 [Expand table](#)

Property	Type	Description
HostingHwnd	UInt64	The handle to the hosting window. This property is used to provide access to the window that hosts the Command Palette.
LanguageOverride	String	The language override for the Command Palette. This property is used to define the language used for localization in the Command Palette.

Methods

 [Expand table](#)

Method	Description
HideStatus(IStatusMessage)	Hides the status message in the Command Palette. This method is used to remove the status message from the Command Palette.
LogMessage(ILogMessage)	Logs a message to the Command Palette. This method is used to provide logging functionality for extensions in the Command Palette.
ShowStatus(IStatusMessage)	Shows a status message in the Command Palette. This method is used to display a status message in the Command Palette.

IExtensionHost.HideStatus(IStatusMessage) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **HideStatus** method is used to hide a status message in the Command Palette. This method is used to remove the status message from the Command Palette, providing a way to clear feedback or notifications that are no longer relevant.

Parameters

message [IStatusMessage](#)

The status message to be hidden in the Command Palette. This parameter is used to define the content of the status message that should be removed.

Returns

A **Windows.Foundation.IAsyncAction** object that represents the asynchronous operation. This method does not return a value, but it can be awaited to ensure that the status message is hidden before proceeding with other operations.

IExtensionHost.LogMessage(ILogMessage) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **LogMessage** method is used to log a message to the Command Palette. This method is used to provide logging functionality for extensions in the Command Palette.

Parameters

message [ILogMessage](#)

The message to be logged in the Command Palette. This parameter is used to define the content of the log message.

Returns

A **Windows.Foundation.IAsyncAction** object that represents the asynchronous operation. This method does not return a value, but it can be awaited to ensure that the log message is processed before proceeding with other operations.

IExtensionHost.ShowStatus(IStatusMessage) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ShowStatus** method is used to display a status message in the Command Palette. This method is used to provide feedback to the user about the current state of the Command Palette or the extension.

Parameters

message [IStatusMessage](#)

The status message to be displayed in the Command Palette. This parameter is used to define the content of the status message.

Returns

A **Windows.Foundation.IAsyncAction** object that represents the asynchronous operation. This method does not return a value, but it can be awaited to ensure that the status message is displayed before proceeding with other operations.

IFallbackCommandItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IFallbackCommandItem** interface represents a fallback command item in the Command Palette. It is used to define the properties and methods associated with a fallback command item that can be displayed in the Command Palette when no other commands match the user's input.

Properties

 [Expand table](#)

Property	Type	Description
FallbackHandler	IFallbackHandler	The fallback handler associated with the command item. This property is used to define the behavior of the command item when it is selected.

IFallbackHandler Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IFallbackHandler** interface is used to handle fallback scenarios in the Command Palette. This is commonly used for commands that want to allow the user to search for something that they've typed that doesn't match any of the commands in the list.

Methods

 Expand table

Method	Description
UpdateQuery(String)	Updates the query string used for fallback handling. This method is called when the user types a query that doesn't match any of the commands in the Command Palette.

IFallbackHandler.UpdateQuery(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **UpdateQuery** method is used to update the query string used for fallback handling. This method is called when the user types a query that doesn't match any of the commands in the Command Palette.

Parameters

query String

The query string that is used for fallback handling. This is the text that the user has typed into the Command Palette.

IFilter Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IFilter** interface represents a filter in the Command Palette. It is used to define the properties and methods associated with a filter that can be applied to the items displayed in the Command Palette.

Properties

 Expand table

Property	Type	Description
Icon	IconInfo	The icon associated with the filter. This property is used to define the visual representation of the filter in the Command Palette.
Id	String	The unique identifier for the filter. This property is used to identify the filter in the Command Palette.
Name	String	The name of the filter. This property is used to define the display name of the filter in the Command Palette.

IFilters Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)


The **IFilters** interface is used to manage the filters in the Command Palette. It provides methods to retrieve the current filter ID and to get a list of available filters.

Properties

 Expand table

Property	Type	Description
CurrentFilterId	String	The ID of the currently selected filter. This property is used to identify which filter is currently active in the Command Palette.

Methods

 Expand table

Method	Description
Filters()	Retrieves a list of available filters in the Command Palette. This method is used to get the filters that can be applied to the items displayed in the Command Palette.

IFilters.Filters() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Filters** method retrieves a list of available filters in the Command Palette. This method is used to get the filters that can be applied to the items displayed in the Command Palette.

Returns

An [IFilterItem\[\]](#) that contains the list of available filters. Each item in the array represents a filter that can be applied to the items displayed in the Command Palette.

IFilterItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IFilterItem** interface represents a filter item in the Command Palette. It is used to define the properties and methods associated with a filter item that can be applied to the items displayed in the Command Palette.

Properties

IForm Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IForm** interface represents a form in the Command Palette. It is used to define the properties and methods for creating and managing forms, such as displaying fields, handling user input, and validating data.

Methods

 Expand table

Method	Description
DataJson()	The JSON representation of the data associated with the form.
StateJson()	The JSON representation of the state associated with the form.
SubmitForm(String)	Submits the form data.
TemplateJson()	The JSON representation of the template associated with the form.

IForm.DataJson() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **DataJson** method retrieves the JSON representation of the data associated with the form. It is used to get the data that defines the structure and content of the Adaptive Card to be displayed to the user.

Returns

A **String** that contains the JSON representation of the data associated with the form.

IForm.StateJson() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **StateJson** method retrieves the JSON representation of the state associated with the form. It is used to get the state data that defines the current status and values of the form fields.

Returns

A **String** that contains the JSON representation of the state associated with the form. This JSON data is typically used to define the current values and status of the form fields in the Adaptive Card that will be displayed in the Command Palette.

IForm.SubmitForm(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **SubmitForm** method submits the form data. It is used to send the data entered in the Adaptive Card to the Command Palette for processing.

Parameters

payload **String**

The JSON representation of the data to be submitted. This data is typically collected from user input in the form fields.

Returns

An [ICommandResult](#) object that contains the result of the form submission.

IForm.TemplateJson() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **TemplateJson** method retrieves the JSON representation of the template associated with the form. It is used to get the template data that defines the structure and layout of the Adaptive Card to be displayed to the user.

Returns

A **String** that contains the JSON representation of the template associated with the form. This JSON data is typically used to define the layout and structure of the Adaptive Card that will be displayed in the Command Palette.

IFormContent Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)


The **IFormContent** interface represents the content of a form in the Command Palette. It is used to define the properties and methods for creating and managing form content, such as displaying data, handling user input, and validating data.

Properties

 Expand table

Property	Type	Description
DataJson	String	The JSON representation of the data associated with the form content.
StateJson	String	The JSON representation of the state associated with the form content.
TemplateJson	String	The JSON representation of the template associated with the form content.

Methods

 Expand table

Method	Description
SubmitForm(String)	Submits the form data.

IFormContent.SubmitForm(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **SubmitForm** method submits the form data. It is used to send the data entered in the form to the Command Palette for processing.

Parameters

payload String

The JSON representation of the data to be submitted. This data is typically collected from user input in the form fields.

Returns

An [ICommandResult](#) object that contains the result of the form submission.

IFormPage Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IFormPage** interface represents a form page in the Command Palette. It is used to define the properties and methods for creating and managing form pages, such as displaying forms, handling user input, and validating data.

Methods

 Expand table

Method	Description
Forms()	Gets the forms associated with the form page.

IFormPage.Forms() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Forms** method retrieves the forms associated with the form page. It is used to get the list of forms that can be displayed in the Command Palette.

Returns

An [IForm\[\]](#) array that contains the forms associated with the form page. The array may be empty if no forms are associated with the page.

IGoToPageArgs Interface

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Derived [GoToPageArgs](#)

The **IGoToPageArgs** interface represents the arguments for navigating to a specific page in the Command Palette. It is used to define the parameters for navigating to a page, such as the page ID and navigation mode.

Properties

 Expand table

Property	Type	Description
NavigationMode	NavigationMode	Gets or sets the navigation mode.
Pageld	String	Gets or sets the ID of the page to navigate to.

IGridProperties Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IGridProperties** interface represents the properties of a grid in the Command Palette. It is used to define the layout and appearance of grids in the Command Palette, such as tile size.

Properties

 Expand table

Property	Type	Description
TileSize	Windows.Foundation.Size	Gets or sets the size of the tiles in the grid.

IconData Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IconData** interface represents the data for an icon that can be used in the Command Palette. It is used to define icons that can be displayed in the Command Palette, such as application icons, file icons, or custom icons.

Properties

 Expand table

Property	Type	Description
Icon	String	Gets the icon.
Data	Windows.Storage.Streams.IRandomAccessStreamReference	Gets the data of the icon.

IconInfo Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IconInfo** interface represents an icon that can be used in the Command Palette. It is used to define icons that can be displayed in the Command Palette, such as application icons, file icons, or custom icons.

Properties

 Expand table

Property	Type	Description
Light	IconData	Gets the light mode version of the icon.
Dark	IconData	Gets the dark mode version of the icon.

IInvokableCommand Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Derived [CopyTextCommand](#)

The **IInvokableCommand** interface represents a command that can be invoked in the Command Palette. It is used to define commands that can be executed by the user, such as opening a file, running a script, or performing an action.

Methods

 Expand table

Method	Description
Invoke(Object)	Invokes the command with the specified arguments.

IInvokableCommand.Invoke(Object) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The method called when a user selects a command.

Parameters

sender Object

Represents the context of where the command was invoked from. This can be different types depending on where the command is being used:

- [TopLevelCommands](#) (and fallbacks): *sender* is the [ICommandItem](#) for the top-level command that was invoked.
- [IListPage.GetItems\(\)](#): *sender* is the [IListItem](#) for the list item selected for that command.
- [ICommandItem.MoreCommands](#) (context menus): *sender* is either the [IListItem](#) which the command was attached to for a list page or the [ICommandItem](#) of the top-level command (if this is a context item on a top-level command).
- [IContentPage.Commands](#): *sender* is the [IContentPage](#) itself.

Using the *sender* parameter can be useful for big lists of items where the actionable information for each item is somewhat the same. One example would be a long list of links. You can implement this as a single [IInvokableCommand](#) that opens a URL based on the *sender* object passed in. Then, each list item would store the URL to open and the title of the link. This creates less overhead for the extension and host to communicate.

Returns

An [ICommandResult](#) object that represents the result of the command invocation. This object can contain information about the success or failure of the command, as well as any additional data that may be relevant to the command's execution.

Example

See [Add a command](#) for a sample of how to implement this method.

ItemsChangedEventArgs Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ItemsChangedEventArgs** interface represents the arguments for the event that is raised when the items in the Command Palette change. It provides information about the number of items that have been added, removed, or modified in the list.

Properties

 Expand table

Property	Type	Description
TotalItems	Int32	Gets the number of items changed.

IListItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IListItem** interface represents an item in the Command Palette list. It is used to define the properties and behavior of individual items displayed in the list, allowing users to interact with and select from the available options.

Properties

 Expand table

Property	Type	Description
Details	IDetails	The details of the item. This property can be used to provide additional information or context about the item, such as a description or metadata.
Section	String	The section to which the item belongs. This property can be used to group items into sections, allowing for better organization and navigation within the list.
Tags	ITag[]	The tags associated with the item. This property can be used to categorize or label the item, making it easier for users to filter and search for specific items.
TextToSuggest	String	The text used to suggest the item. This property can be used to provide a hint or suggestion to the user about the item, helping them to understand its purpose or functionality.

IListPage Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)


The **IListPage** interface represents a page in the Command Palette that displays a list of items. It is used to present a collection of items in a structured format, allowing users to interact with and select from the list.

Properties

 Expand table

Property	Type	Description
EmptyContent	ICommandItem	The content to display when the list is empty. This property can be used to provide a message or action for the user when there are no items to display.
Filters	IFilters	The filters applied to the list. This property allows users to refine the items displayed in the list based on specific criteria.
GridProperties	IGridProperties	The properties of the grid layout used to display the list items. This property defines how the items are arranged and presented in the grid.
HasMoreItems	Boolean	Indicates whether there are more items to load in the list. This property is used to determine if additional items can be fetched or displayed.
PlaceholderText	String	The text to display when the list is empty or no items match the current filters. This property provides context to the user about the state of the list.
SearchText	String	The text used to filter the list items. This property allows users to search for specific items within the list.
ShowDetails	Boolean	Indicates whether to show detailed information about the items in the list. This property can be used to toggle between a summary view and a detailed view of the items.

Methods

 Expand table

Method	Description
GetItems()	Retrieves the items to be displayed in the list. This method is used to fetch the data that populates the list.
LoadMore()	Loads more items into the list. This method is used to fetch additional data when the user requests more items, such as when scrolling or clicking a "Load More" button.

IListPage.GetItems() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **GetItems** method retrieves the items to be displayed in the list. This method is used to fetch the data that populates the list.

Returns

An [IListItem\[\]](#) containing the items to be displayed in the list. The array may be empty if there are no items to display or if the filters applied to the list result in no matching items.

IListPage.LoadMore() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **LoadMore** method is used to load additional items into the list. This method is typically called when the user requests more items, such as when scrolling to the end of the list or clicking a "Load More" button.

ILogMessage Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ILogMessage** interface represents a log message in the Command Palette. It is used to provide information about the state of the application or specific operations, and can be used for debugging or informational purposes.

Properties

 Expand table

Property	Type	Description
Message	String	The message text that describes the log entry.
State	MessageState	The state of the log message. This property indicates the severity or type of the log message, such as informational, warning, or error.

IMarkdownContent Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IMarkdownContent** interface represents a content element in the Command Palette that can display Markdown-formatted text. It is used to provide rich text formatting and layout options for displaying information in the Command Palette.

Properties

 Expand table

Property	Type	Description
Body	String	The body of the content, formatted using Markdown. This property contains the main text content that will be displayed in the Command Palette.

IMarkdownPage Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IMarkdownPage** interface represents a page in the Command Palette that can display Markdown content. It is used to provide rich text formatting and layout options for displaying information in the Command Palette.

Properties

 Expand table

Property	Type	Description
Commands	IContextItem[]	The commands associated with the page.

Methods

 Expand table

Method	Description
Bodies()	Returns the bodies of the page.
Details()	Returns the details of the page.

IMarkdownPage.Bodies() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Bodies** method returns the bodies of the page. This method is used to retrieve the main content of the page, which can include text, images, and other elements formatted using Markdown.

Returns

A **String[]** that contains the bodies of the page. Each element in the array represents a separate body of content, which can be displayed in the Command Palette using Markdown formatting. The bodies may include headings, paragraphs, lists, images, and other Markdown elements.

IMarkdownPage.Details() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **Details** method returns the details of the page. This method is used to retrieve additional information about the page, such as metadata or configuration settings.

Returns

An [IDetails](#) object that contains the details of the page. The details may include information such as the page's title, description, and other relevant metadata.

INotifyItemsChanged Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **INotifyItemsChanged** interface represents an object that can notify listeners of item changes in the Command Palette. It is used to implement the observer pattern, allowing other components to subscribe to item change notifications.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<Object, IItemsChangedEventArgs > ItemsChanged	Notifies that items have changed.

INotifyPropertyChanged Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **INotifyPropertyChanged** interface represents an object that can notify listeners of property changes in the Command Palette. It is used to implement the observer pattern, allowing other components to subscribe to property change notifications.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<Object, IPropertyChangedEventArgs > PropertyChanged	Notifies that a property value has changed.

IPage Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Represents individual views in the application and are the primary unit of navigation and interaction.

Properties

 Expand table

Property	Type	Description
AccentColor	OptionalColor	The accent color of the page.
IsLoading	Boolean	Indicates whether the page is currently loading.
Title	String	The title of the page.

IProgressState Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IProgressState** interface represents the state of a progress indicator in the Command Palette. It is used to provide information about the progress of an operation, such as loading or processing data.

Properties

 Expand table

Property	Type	Description
IsIndeterminate	Boolean	Indicates whether the progress is indeterminate (true) or determinate (false).
ProgressPercent	UInt32	Gets the progress percentage (0 - 100) if the progress is determinate.

IPropChangedEventArgs Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IPropChangedEventArgs** interface represents the arguments for property change events in the Command Palette. It is used to provide information about the property that has changed and its new value.

Properties

 Expand table

Property	Type	Description
PropertyName	String	Gets the property name.

ISeparatorContextItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ISeparatorContextItem** interface represents a separator context menu item in the Command Palette. It is used to create a visual separation between different sections of commands or items in the Command Palette.

Properties

ISeparatorFilterItem Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ISeparatorFilterItem** interface represents a separator filter item in the Command Palette. It is used to create a visual separation between different sections of commands or items in the Command Palette.

Properties

IStatusMessage Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **IStatusMessage** interface is used to define a status message in the Command Palette. It provides properties to specify the message text, progress state, and overall state of the message.

Properties

 Expand table

Property	Type	Description
Message	String	The message text to be displayed in the Command Palette. This property is used to provide information or feedback to the user.
Progress	IProgressState	The progress state of the message. This property is used to indicate the progress of an operation or task associated with the message.
State	MessageState	The overall state of the message. This property is used to categorize the message, such as success, error, or warning.

ITag Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ITag** interface represents a tag in the Command Palette. Tags are used to categorize and organize commands, making them easier for users to find and execute.

Properties

 Expand table

Property	Type	Description
Background	OptionalColor	The background color of the tag. This property is used to visually distinguish the tag from other elements in the Command Palette.
Foreground	OptionalColor	The foreground color of the tag. This property is used to set the text color of the tag, ensuring good contrast with the background color.
Icon	IIconInfo	The icon associated with the tag. This property is used to provide a visual representation of the tag, making it easier for users to identify its purpose.
Text	String	The text displayed on the tag. This property is used to provide a label for the tag, indicating its category or purpose.
ToolTip	String	The tooltip text displayed when the user hovers over the tag. This property is used to provide additional information about the tag, helping users understand its purpose.

IToastArgs Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

Derived [ToastArgs](#)

The **IToastArgs** interface is used to define the arguments for a toast notification in the Command Palette. It provides properties to specify the message and the result of the command associated with the toast.

Properties

 [Expand table](#)

Property	Type	Description
Message	String	The message to be displayed in the toast notification.
Result	ICommandResult	The result of the command associated with the toast notification. This property is used to provide additional information about the command's execution.

ITreeContent Interface


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)


The **ITreeContent** interface is used to represent a tree structure of content on a content page in the Command Palette. It allows for hierarchical organization of content, enabling users to navigate through different levels of information.

Properties

 Expand table

Property	Type	Description
RootContent	IContent	Gets the root content of the tree. This is the top-level item in the hierarchy.

Methods

 Expand table

Method	Description
GetChildren()	Retrieves the child items of the current content. This method is used to navigate through the tree structure.

KeyChord Struct

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **KeyChord** struct represents a combination of key modifiers and a virtual key. It is used to define keyboard shortcuts in the Command Palette.

Fields

 Expand table

Field	Type	Description
Modifiers	Windows.System.VirtualKeyModifiers	The key modifiers (e.g., Control, Shift, Alt) that are part of the key chord.
Vkey	Int32	The virtual key code that represents the main key in the key chord.
ScanCode	Int32	The scan code that represents the physical key on the keyboard. This is used for low-level keyboard input handling.

MessageState Enum


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **MessageState** enum defines the different states of a message that can be displayed in the Command Palette. Each state represents a different type of message that can be shown to the user, such as error messages, informational messages, success messages, and warning messages.

Fields

 Expand table

Field	Value	Description
Info	0	Indicates an informational message.
Success	1	Indicates a success message.
Warning	2	Indicates a warning message.
Error	3	Indicates an error message.

NavigationMode Enum

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **NavigationMode** enum defines the different modes of navigation that can be used in the Command Palette. Each mode specifies how the navigation stack should be managed when navigating to a new page. This is useful for controlling the behavior of the navigation experience in your application.

Fields

 [Expand table](#)

Field	Description
GoBack	Go back one level, then navigate to the page. Going back from the requested page will take you to the page before the current page.
GoHome	Clear the back stack, then navigate to the page. Going back from the requested page will take you to the home page.
Push	The new page gets added to the current navigation stack. Going back from the requested page will take you to the current page.

OptionalColor Struct

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **OptionalColor** struct represents an optional color value that can be used in the Command Palette. It is designed to encapsulate a color value along with a flag indicating whether the value is present or not. This is useful for scenarios where a color may or may not be specified, allowing for more flexible and dynamic UI designs.

Fields

 [Expand table](#)

Field	Type	Description
Color	Microsoft.CommandPalette.Extensions.Color	The color value. This field is used to store the actual color when it is specified.
HasValue	Boolean	Indicates whether the color value is present or not.

ProviderType Enum


Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions](#)

The **ProviderType** enum defines the types of providers that can be used in the Command Palette. Each provider type represents a different category of functionality or data source that can be integrated into the Command Palette.

Fields

 Expand table

Name	Value	Description
Commands		Represents a provider that offers commands. Commands are actions that can be executed directly from the Command Palette.

Microsoft.CommandPalette.Extensions.Toolkit Namespace

Article • 03/31/2025

Contains helper classes that make creating extensions easier.

Interfaces

 Expand table

Interface	Description
ISettingsForm	Defines the interface for a settings form.

Classes

 Expand table

Class	Description
AnonymousCommand	AnonymousCommand is a command that can be invoked without a specific target. It is typically used for commands that do not require any parameters or context.
BaseObservable	Base class for observable objects.
ChoiceSetSetting	Represents a setting that allows the user to choose from a set of predefined options.
ClipboardHelper	Helper class for clipboard operations.
ColorHelpers	Helper class for color operations.
Command	Base class for commands.
CommandContextItem	Represents an item in the command's context menu.
CommandItem	Represents an item in the command palette.
CommandProvider	Base class for command providers.
CommandResult	Represents the result of a command invocation.
ConfirmationArgs	Represents the arguments for a confirmation dialog.

Class	Description
ContentPage	Represents a page that contains content.
CopyTextCommand	Represents a command that copies text to the clipboard.
Details	Represents a details view in the command palette.
DetailsCommand	Represents a command that displays details in the command palette.
DetailsElement	Represents an element in the details view.
DetailsLink	Represents a link in the details view.
DetailsSeparator	Represents a separator in the details view.
DetailsTags	Represents tags in the details view.
DynamicListPage	Represents a dynamic list page in the command palette.
ExtensionHost	Represents the host for an extension.
FallbackCommandItem	Represents a fallback command item in the command palette.
Filter	Represents a filter for command items.
FormContent	Represents a form in the command palette.
GoToPageArgs	Represents the arguments for navigating to a page.
IconData	Represents icon data for command items.
IconHelpers	Helper class for icon operations.
IconInfo	Represents information about an icon.
InvokableCommand	Represents a command that can be invoked.
ItemsChangedEventArgs	Represents the event arguments for items changed events.
JsonSettingsManager	Manages settings stored in JSON format.
KeyChordHelpers	Helper class for key chord operations.
ListHelpers	Helper class for list operations.
ListItem	Represents an item in a list.
ListPage	Represents a list page in the command palette.
LogMessage	Represents a log message.
MarkdownContent	Represents markdown content in the command palette.

Class	Description
MatchOption	Represents options for matching command items.
MatchResult	Represents the result of a match operation.
NoOpCommand	Represents a command that does nothing. It is typically used as a placeholder.
OpenUrlCommand	Represents a command that opens a URL in the default web browser.
Page	Base class for pages in the command palette.
ProgressState	Represents the state of a progress operation.
PropChangedEventArgs	Represents the event arguments for property changed events.
Setting	Base class for settings.
Settings	Represents the settings for an extension.
SettingsForm	Represents a form for displaying settings.
ShellHelpers	Helper class for shell operations.
StatusMessage	Represents a status message in the command palette.
StringMatcher	Helper class for string matching operations.
Tag	Represents a tag in the command palette.
TextSetting	Represents a setting that allows the user to enter text.
ThumbnailHelper	Helper class for thumbnail operations.
ToastArgs	Represents the arguments for a toast notification.
ToastStatusMessage	Represents a toast status message in the command palette.
ToggleSetting	Represents a setting that allows the user to toggle a value on or off.
TreeContent	Represents tree content in the command palette.
Utilities	Utility class for common operations.

Enums

 Expand table

Enum	Description
SearchPrecisionScore	The precision score of a search result.

AnonymousCommand Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [InvokableCommand](#)


The **AnonymousCommand** class is a command that can be invoked without being associated with a specific command palette item. It is typically used for commands that do not require any parameters or additional context.

Constructors

 Expand table


Constructor	Description
AnonymousCommand(Action)	Initializes a new instance of the AnonymousCommand class with the specified action.

Properties

 Expand table

Property	Type	Description
Result	ICommandResult	Gets the result of the command execution.

Methods

 Expand table

Method	Description
Invoke()	Invokes the command.

AnonymousCommand Constructors

Article • 03/31/2025

AnonymousCommand(Action) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [AnonymousCommand](#) class with an *action* parameter.

C#

```
public AnonymousCommand(Action? action)
{
    Name = "Invoke";
    _action = action;
}
```

Parameters

action **Action**

The action to be executed when the command is invoked. This parameter is optional and can be null.

AnonymousCommand.Invoke() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Invoke** method executes the command associated with the **AnonymousCommand** instance. This method does not require any parameters and can be called directly to trigger the command's action.

Returns

An [ICommandResult](#) object that represents the result of the command execution. The result may contain information about the success or failure of the command, as well as any relevant data returned by the command's action.

BaseObservable Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [INotifyPropertyChanged](#)

The **BaseObservable** class is a base class for objects that need to notify clients about changes to their properties. It implements the **INotifyPropertyChanged** interface and provides a **PropertyChanged** event.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<object, IPropertyChangedEventArgs> PropertyChanged	Notifies that a property value has changed.

Methods

 Expand table

Method	Description
OnPropertyChanged(String)	Raises the PropertyChanged event to notify that a property value has changed.

BaseObservable.OnPropertyChanged(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Raises the [PropChanged](#) event to notify that a property value has changed.

Parameters

propertyName **String**

The name of the property that has changed.

ChoiceSetSetting Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Setting<String>](#)


The **ChoiceSetSetting** class represents a setting that allows users to select from a predefined set of choices. This is useful for applications that need to provide users with a limited set of options to choose from, such as selecting a theme or a configuration option.

Constructors

 Expand table


Constructor	Description
ChoiceSetSetting(String, List<Choice>)	Initializes a new instance of the ChoiceSetSetting class with a specified name and a list of choices.
ChoiceSetSetting(String, String, String, List<Choice>)	Initializes a new instance of the ChoiceSetSetting class with a specified name, description, default value, and a list of choices.

Nested classes

 Expand table


Class	Description
Choice	Represents a single choice in the choice set. Each choice has a name and a value. The name is displayed to the user, while the value is used internally by the application.

Properties

 Expand table

Property	Type	Description
Choices	List< Choice >	Gets or sets the list of choices available in the choice set. Each choice is represented by an instance of the Choice class.

Methods

 Expand table

Method	Description
LoadFromJson(JsonObject)	Loads the setting from a JSON object. This is useful for applications that need to deserialize settings from a configuration file or other JSON source.
ToDictionary()	Converts the setting to a dictionary representation. This is useful for applications that need to serialize settings to a format that can be easily stored or transmitted.
ToState()	Converts the setting to a state representation. This is useful for applications that need to represent settings in a format that can be easily displayed or manipulated.
Update(JsonObject)	Updates the setting from a JSON object. This is useful for applications that need to apply changes to settings based on user input or other sources.

ChoiceSetSetting Constructors

Article • 03/31/2025

ChoiceSetSetting(String, List<Choice>) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ChoiceSetSetting](#) class from the base [Setting](#) class, setting its [Key](#) property to *key* and its [Choices](#) set to *choices*.

C#

```
public ChoiceSetSetting(string key, List<Choice> choices)
    : base(key, choices.ElementAt(0).Value)
{
    Choices = choices;
}
```

Parameters

key String

The key for the setting. This is a unique identifier that is used to reference the setting in the application.

choices List<[Choice](#)>

The list of choices for the setting. Each choice is represented by a [Choice](#) object, which contains the display text and value for the choice. The first choice in the list is used as the default value for the setting.

ChoiceSetSetting(String, String, String, List<Choice>) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ChoiceSetSetting](#) class from the base [Setting](#) class, setting its [Key](#) property to *key*, its [Label](#) to *label*, its [Description](#) to *description*, and its [Choices](#) to *choices*.

C#

```
public ChoiceSetSetting(string key, string label, string description,
    List<Choice> choices)
    : base(key, label, description, choices.ElementAt(0).Value)
{
    Choices = choices;
}
```

Parameters

key String

The key for the setting. This is a unique identifier that is used to reference the setting in the application.

label String

The label for the setting. This is a user-friendly name that is displayed to the user in the application.

description String

The description for the setting. This provides additional information about the setting and its purpose.

choices List<[Choice](#)>

The list of choices for the setting. Each choice is represented by a [Choice](#) object, which contains the display text and value for the choice. The first choice in the list is used as the default value for the setting.

Choice Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Choice** class represents a single choice in a choice set. It contains properties for the display text and value of the choice.

Choice(String, String) Constructor

Definition

Initializes a new instance of the `Choice` class with the given `title` and `value`.

C#

```
public Choice(string title, string value)
{
    Value = value;
    Title = title;
}
```

Parameters


title String

The display text for the choice. This is the text that will be shown to the user in the UI.

value String

The value for the choice. This is the value that will be returned when the choice is selected. It can be any string that represents the choice in a meaningful way.

Properties

 Expand table

Property	Type	Description
Title	String	The display text for the choice.
Value	String	The value for the choice.

ChoiceSetSetting.LoadFromJson(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LoadFromJson** method loads the setting from a JSON object. This is useful for applications that need to initialize settings from a configuration file or other sources.

Parameters

jsonObject **JsonObject**

The **JsonObject** that contains the values for the setting. This object should include the properties that need to be set, such as the name, description, default value, and list of choices.

Returns

A [ChoiceSetSetting](#) object that represents the loaded setting.

ChoiceSetSetting.ToDictionary() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDictionary** method converts the setting to a dictionary representation. This is useful for applications that need to serialize settings to a format that can be easily stored or transmitted.

Returns

A **Dictionary<String, Object>** that represents the setting. This dictionary can be used to store the setting in a format that can be easily serialized or transmitted, such as JSON or XML.

ChoiceSetSetting.ToState() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToState** method converts the setting to a state representation. This is useful for applications that need to represent settings in a format that can be easily displayed or manipulated.

Returns

A **String** that represents the state of the setting. This string can be used to display the current value of the setting to the user or to perform other operations based on the state of the setting.

ChoiceSetSetting.Update(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the setting from a JSON object. This is useful for applications that need to apply changes to settings based on user input or other sources.

Parameters

payload **JsonObject**

The **JsonObject** that contains the updated values for the setting. This object should include the properties that need to be updated, such as the name, description, default value, and list of choices.

ClipboardHelper Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ClipboardHelper** class provides methods for interacting with the clipboard. It allows you to get and set text and RTF (Rich Text Format) data on the clipboard. This is useful for applications that need to manipulate clipboard data, such as copying and pasting formatted text.

Methods

 Expand table

Method	Description
GetText()	Gets the text data from the clipboard.
SetRtf(String, String)	Sets the RTF data on the clipboard.
SetText(String)	Sets the text data on the clipboard.

ClipboardHelper.GetText() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **GetText** method retrieves the text data from the clipboard. This is useful for applications that need to access clipboard data, such as pasting text into a text box or other UI element.

Returns

A **String** that represents the text data retrieved from the clipboard. This can be any string value, including plain text or formatted text. If the clipboard does not contain text data, this method may return an empty string or null.

ClipboardHelper.SetRtf(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SetRtf** method sets the RTF (Rich Text Format) data on the clipboard. This is useful for applications that need to copy formatted text to the clipboard for pasting into other applications.

Parameters

plainText **String**

The plain text representation of the RTF data. This is the text that will be displayed when the RTF data is pasted into applications that do not support RTF.

rtfText **String**

The RTF data to set on the clipboard. This is the formatted text that will be pasted into applications that support RTF. The RTF data should be a valid RTF string, including formatting information such as font styles, colors, and sizes.

ClipboardHelper.SetText(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SetText** method sets the text data on the clipboard. This is useful for applications that need to copy text to the clipboard for pasting into other applications.

Parameters

text **String**

The text to set on the clipboard. This can be any string value, including plain text or formatted text.

ColorHelpers Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ColorHelpers** class provides static methods for creating colors in the Command Palette SDK. It is used to create colors that can be used in various UI elements, such as command items and icons.

Methods

 Expand table

Method	Description
FromArgb(Byte, Byte, Byte, Byte)	Creates a color from the specified ARGB values.
FromRgb(Byte, Byte, Byte)	Creates a color from the specified RGB values.
NoColor()	Creates a color that represents no color.
Transparent()	Creates a color that represents transparency.

ColorHelpers.FromArgb(Byte, Byte, Byte, Byte) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FromArgb** method creates a color from the specified ARGB values. This is useful when you want to specify a color using its alpha, red, green, and blue components.

Parameters

a Byte

The alpha component of the color. This value should be between `0` and `255`, where `0` is fully transparent and `255` is fully opaque.

r Byte

The red component of the color. This value should be between `0` and `255`.

g Byte

The green component of the color. This value should be between `0` and `255`.

b Byte

The blue component of the color. This value should be between `0` and `255`.

Returns

An [OptionalColor](#) object that represents the color created from the specified ARGB values. This object can be used in various UI elements, such as command items and icons, to specify the color of the element.

ColorHelpers.FromRgb(Byte, Byte, Byte) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FromRgb** method creates a color from the specified RGB values. This is useful when you want to specify a color using its red, green, and blue components.

Parameters

r Byte

The red component of the color. This value should be between 0 and 255.

g Byte

The green component of the color. This value should be between 0 and 255.

b Byte

The blue component of the color. This value should be between 0 and 255.

Returns

An [OptionalColor](#) object that represents the color created from the specified RGB values. This object can be used in various UI elements, such as command items and icons, to specify the color of the element.

ColorHelpers.NoColor() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **NoColor** method creates a color that represents no color. This is useful when you want to specify a color that should not be visible, such as when you want to hide an element or make it invisible.

Returns

An [OptionalColor](#) object that represents no color.

ColorHelpers.Transparent() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Transparent** method creates a color that represents transparency. This is useful when you want to specify a color that should not be visible, such as when you want to hide an element or make it invisible.

Returns

An [OptionalColor](#) object that represents transparency. This object can be used in various UI elements, such as command items and icons, to indicate that the element should not be visible.

Command Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [ICommand](#)

Commands are the primary unit of functionality in the Command Palette SDK. They represent "a thing that a user can do". These can be something simple like open a URL in a web browser. Or they can be more complex, with nested commands, custom arguments, and more.

Properties

 Expand table

Property	Type	Description
Icon	IconInfo	Gets or sets the icon for the command.
Id	String	Gets or sets the ID of the command.
Name	String	Gets or sets the name of the command.

CommandContextItem Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [CommandItem](#)

Implements [ICommandContextItem](#)


The **CommandContextItem** class represents a command item in the command palette that is associated with a specific context. It extends the functionality of the **CommandItem** class by adding properties and behaviors related to context-specific commands. This class is used to create and manage command items that are relevant to a particular context within the command palette framework.

Constructors

 Expand table

Constructor	Description
CommandContextItem(ICommand)	Initializes a new instance of the CommandContextItem class, setting its Command property to <i>command</i> and its Title to the Name of the <i>command</i> .
CommandContextItem(String, String, String, Action, ICommandResult)	Initializes a new instance of the CommandContextItem class, setting its Title , Subtitle , Icon , and Command properties.

Properties

 Expand table

Property	Type	Description
IsCritical	Boolean	Indicates whether the command item is critical.
RequestedShortcut	KeyChord	The requested keyboard shortcut for the command item.

CommandContextItem Constructors

Article • 03/31/2025

CommandContextItem(ICommand) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandContextItem](#) class from the base [CommandItem](#) class, setting its [Command](#) property to *command*.

C#

```
public CommandContextItem(ICommand command)
    : base(command)
{
}
```

Parameters

command [ICommand](#)

The command to be associated with the command item. This parameter is required and cannot be null.

CommandContextItem(String, String, String, Action, ICommandResult) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandContextItem](#) class from the base [CommandItem](#) class, setting its [Title](#) property to *title*, its [Subtitle](#) to *subtitle*, and creates a new [AnonymousCommand](#) object with a *name*, *action*, and *result*.

C#

```

public CommandContextItem(
    string title,
    string subtitle = "",
    string name = "",
    Action? action = null,
    ICommandResult? result = null)
{
    var c = new AnonymousCommand(action);
    if (!string.IsNullOrEmpty(name))
    {
        c.Name = name;
    }

    if (result != null)
    {
        c.Result = result;
    }

    Command = c;

    Title = title;
    Subtitle = subtitle;
}

```

Parameters

title String

The title of the command item.

subtitle String

The subtitle of the command item.

name String

The name of the command item.

action Action

The action to be performed when the command item is executed.

result ICommandResult

The result of the command item execution. This parameter is optional and can be `null`.

CommandItem Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [ICommandItem](#)


The **CommandItem** class represents a command item in the command palette. It encapsulates the properties and behaviors of a command, including its title, subtitle, icon, and associated command logic. This class is used to create and manage command items within the command palette framework.

Constructors

 Expand table

Constructor	Description
CommandItem()	Creates a new instance of the CommandItem class.
CommandItem(ICommand)	Creates a new instance of the CommandItem class with the specified command.
CommandItem(ICommandItem)	Creates a new instance of the CommandItem class with the specified command item.
CommandItem(String, String, String, Action, ICommandResult)	Creates a new instance of the CommandItem class with the specified title, subtitle, icon, command action, and command result.

Properties

 Expand table

Property	Type	Description
Command	ICommand	The command associated with the command item. This property allows access to the command's logic and execution

Property	Type	Description
		behavior.
Icon	IIconInfo	The icon associated with the command item. This property defines the visual representation of the command in the command palette.
MoreCommands	IContextItem[]	An array of commands associated with the command item. This property defines the context menu on the bottom right of the Command Palette.
Subtitle	String	The subtitle of the command item. This property provides additional context or information about the command, enhancing the user experience.
Title	String	The title of the command item. This property represents the primary label or name of the command, displayed in the command palette.

CommandItem Constructors

Article • 03/31/2025

CommandItem() Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandItem](#) class.

C#

```
public CommandItem()  
    : this(new NoOpCommand())  
{  
}
```

CommandItem(ICommand) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandItem](#) class, setting its [Command](#) property to *command* and its [Title](#) to the *command's Name*.

C#

```
public CommandItem(ICommand command)  
{  
    Command = command;  
    Title = command.Name;  
}
```

Parameters

command [ICommand](#)

The command associated with the command item. This property allows access to the command's logic and execution behavior.

CommandItem(ICommandItem) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandItem](#) class, setting its [Command](#) property to *other's* [Command](#), its [Title](#) to *other's* [Title](#), its [Subtitle](#) to *other's* [Subtitle](#), its [Icon](#) to *other's* [Icon](#), and its [MoreCommands](#) to *other's* [MoreCommands](#).

C#

```
public CommandItem(ICommandItem other)
{
    Command = other.Command;
    Title = other.Title;
    Subtitle = other.Subtitle;
    Icon = (IconInfo?)other.Icon;
    MoreCommands = other.MoreCommands;
}
```

Parameters

other [ICommandItem](#)

The command item to copy. This parameter is used to initialize the new command item with the properties of an existing command item.

CommandItem(String, String, String, Action, ICommandResult) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CommandItem](#) class, setting its [Title](#) property to *title*, its [Subtitle](#) to *subtitle*, and creates a new [AnonymousCommand](#) object with a *name*, *action*, and *result*.

C#

```
public CommandItem(  
    string title,  
    string subtitle = "",  
    string name = "",  
    Action? action = null,  
    ICommandResult? result = null)  
{  
    var c = new AnonymousCommand(action);  
    if (!string.IsNullOrEmpty(name))  
    {  
        c.Name = name;  
    }  
  
    if (result != null)  
    {  
        c.Result = result;  
    }  
  
    Command = c;  
  
    Title = title;  
    Subtitle = subtitle;  
}
```

Parameters

title String

The title of the command item. This property represents the primary label or name of the command, displayed in the command palette.

subtitle String

The subtitle of the command item. This property provides additional context or information about the command, enhancing the user experience.

name String

The name of the command. This property is used to identify the command within the command palette.

action Action

The action to be performed when the command is executed. This property defines the logic or behavior associated with the command.

result [ICommandResult](#)

The result of the command execution. This property provides information about the outcome of the command, such as success or failure, and any relevant data returned by the command.

CommandProvider Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [ICommandProvider](#)

The **CommandProvider** class is a base class for creating command providers in the Command Palette. It provides a set of properties and methods to manage commands, including top-level commands and fallback commands. The class also supports event handling for changes in command items.

Properties

 Expand table

Property	Type	Description
DisplayName	String	The display name of the command provider.
Frozen	Boolean	Indicates whether the command provider is frozen. A frozen provider cannot be modified.
ICommandProvider.Icon	IconInfo	The icon associated with the command provider.
Icon	IconInfo	The icon associated with the command provider.
Id	String	The unique identifier of the command provider.
Settings	ICommandSettings	The settings associated with the command provider.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<object, ItemsChangedEventArgs > ItemsChanged	Invoked when the command provider's items change.

Methods

 Expand table

Method	Description
Dispose()	Releases the resources used by the command provider.
FallbackCommands()	Returns the fallback commands for the command provider.
GetCommand(String)	Retrieves a command by its ID.
InitializeWithHost(IExtensionHost)	Initializes the command provider with the specified host.
RaiseItemsChanged(Int)	Raises the ItemsChanged event with the specified change type.
TopLevelCommands()	Returns the top-level commands for the command provider.

CommandProvider.Dispose() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Disposes the command provider and releases any resources it holds.

CommandProvider.FallbackCommands() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FallbackCommands** method retrieves a collection of fallback command items.

Returns

An [IFallbackCommandItem\[\]](#) array representing the fallback command items. This array contains the commands that are used as fallback options when no other commands are available or applicable in the command palette context.

CommandProvider.GetCommand(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **GetCommand** method retrieves a command by its unique identifier. This method is used to access a specific command within the command provider, allowing for operations such as executing or updating the command.

Parameters

id String

The unique identifier of the command to retrieve. This identifier is used to locate the specific command within the command provider's collection of commands.

Returns

An [ICommand](#) object representing the command with the specified identifier. If no command with the specified identifier exists, this method returns null.

CommandProvider.InitializeWithHost([IExtensionHost](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **InitializeWithHost** method initializes the command provider with the specified host instance. This method is typically called when the command provider is being registered with the command palette, allowing it to access the host's services and capabilities.

Parameters

host [IExtensionHost](#)

The host instance that provides access to the command palette's functionality. This parameter is used to initialize the command provider with the host's capabilities and services.

CommandProvider.RaiseItemsChanged() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **RaiseItemsChanged** method raises the **ItemsChanged** event with the specified change type. This method is typically used to notify the Command Palette of changes in the command provider's items, such as when new commands are added or existing commands are removed.

CommandProvider.TopLevelCommands() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **TopLevelCommands** method returns the top-level commands for the command provider. These commands are typically displayed in the command palette and can be executed by the user.

Returns

An [ICommandItem\[\]](#) array containing the top-level commands for the command provider. If there are no top-level commands, an empty array is returned.

CommandResult Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Indicates what the Command Palette should do after a command is executed. This allows commands to control the flow of the palette.

Implements [ICommandResult](#)

The **CommandResult** class is used to specify the result of a command execution in the Command Palette. It provides various methods to control the behavior of the Command Palette after a command is executed. This class is useful for managing navigation, displaying messages, and controlling the state of the Command Palette.

Properties

 Expand table

Property	Type	Description
Args	ICommandResultArgs	Gets or sets the arguments associated with the command result. This can include additional data needed for the command result.
Kind	CommandResultKind	Gets or sets the result of the command. Defaults to CommandResultKind.Dismiss .

Methods

 Expand table

Method	Description
Confirm(ConfirmationArgs)	Display a confirmation dialog to the user.
Dismiss()	Close the Command Palette after the action is executed and dismiss the current state. On the next launch, the Command Palette will start from the main page with a blank query.

Method	Description
GoBack()	Navigate to the previous page, and keep it open.
GoHome()	Navigate back to the main page of the Command Palette and keep it open. This clears out the current stack of pages, but keeps the palette open.
GoToPage(GoToPageArgs)	Navigate to a different page in the palette. The GoToPageArgs will specify which page to navigate to.
Hide()	Creates a new CommandResult instance with Kind set to CommandResultKind.Hide and Args set to <code>null</code> .
KeepOpen()	Do nothing. This leaves the palette in its current state, with the current page stack and query.
ShowToast(String)	Display a transient desktop-level message to the user. Creates a new CommandResult with Args set to a new ToastArgs object with its Message set to String .
ShowToast(ToastArgs)	Display a transient desktop-level message to the user. Creates a new CommandResult instance with Args set to ToastArgs .

Example

See [Command Results](#) for an example of how to use this.

CommandResult.Confirm(ConfirmationArgs) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Display a confirmation dialog to the user. The [ConfirmationArgs](#) will specify the title, and description for the dialog. The primary button of the dialog will activate the [Command](#). If [IsPrimaryCommandCritical](#) is `true`, the primary button will be red, indicating that it is a destructive action.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.Confirm](#) and its [Args](#) set to *args*.

Parameters

args [ConfirmationArgs](#)

The arguments for the command. This should be an instance of [ConfirmationArgs](#) that specifies the title and description for the dialog.

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.Dismiss() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Close the Command Palette after the action is executed and dismiss the current state. On the next launch, the Command Palette will start from the main page with a blank query.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.Dismiss](#).

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.GoBack() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Navigate to the previous page, and keep it open.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.GoBack](#) and its [Args](#) set to `null`.

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.GoHome() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Navigate back to the main page of the Command Palette and keep it open. This clears out the current stack of pages, but keeps the palette open.

ⓘ Note

If the command navigates to another application, the palette's default behavior is to hide itself when it loses focus. That will behave the same as [Dismiss](#) in that case.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.GoHome](#) and its [Args](#) set to `null`.

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.GoToPage(GoToPageArgs) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Navigate to a different page in the palette. The [GoToPageArgs](#) will specify which page to navigate to.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.GoToPage](#) and its [Args](#) set to *args*.

Parameters

args [GoToPageArgs](#)

The arguments for the command. This should be an instance of [GoToPageArgs](#) that specifies the page to navigate to.

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.Hide() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.Hide](#) and its [Args](#) set to `null`.

Returns

A [CommandResult](#) instance.

Example

See [Command Results](#) for an example of how to use this.

CommandResult.KeepOpen() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Do nothing. This leaves the palette in its current state, with the current page stack and query.

ⓘ Note

If the command navigates to another application, the palette's default behavior is to hide itself when it loses focus. When the user next activates the Command Palette, it will be in the same state as when it was hidden.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.KeepOpen](#) and its [Args](#) set to `null`.

Returns

A [CommandResult](#) instance.

CommandResult.ShowToast(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Display a transient desktop-level message to the user. This is especially useful for displaying confirmation that an action took place when the palette will be closed. Consider the [CopyTextCommand](#) in the helpers - this command will show a toast with the text "Copied to clipboard", then dismiss the palette.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.ShowToast](#) and its [Args](#) set to a new [ToastArgs](#) object with its [Message](#) set to *message*.

Parameters

message **String**

The message to be displayed in the toast. This should be a brief and informative message that conveys the result of the action taken by the user.

Returns

A [CommandResult](#) instance.

CommandResult.ShowToast(ToastArgs) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Display a transient desktop-level message to the user. This is especially useful for displaying confirmation that an action took place when the palette will be closed. Consider the [CopyTextCommand](#) in the helpers - this command will show a toast with the text "Copied to clipboard", then dismiss the palette.

Creates a new [CommandResult](#) instance with its [Kind](#) property set to [CommandResultKind.ShowToast](#) and its [Args](#) set to *args*.

Parameters

args [ToastArgs](#)

The arguments for the toast message. This includes the message to be displayed and any other relevant information.

Returns

A [CommandResult](#) instance.

ConfirmationArgs Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IConfirmationArgs](#)

The **ConfirmationArgs** class is used to create a confirmation dialog in the Command Palette. It allows you to specify the details for the confirmation dialog. This class is useful when you want to prompt the user for confirmation before executing a potentially destructive action.

Properties

 Expand table

Property	Type	Description
Description	String	Gets or sets the description of the confirmation dialog.
IsPrimaryCommandCritical	Boolean	Gets or sets if the primary command is critical.
PrimaryCommand	ICommand	Gets or sets the primary action command to be executed when the user confirms the dialog.
Title	String	Gets or sets the title of the confirmation dialog.

ContentPage Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Page](#)

Implements [IContentPage](#)

The **ContentPage** class is a specialized page that can be used to display a collection of commands and details in the command palette. It provides a way to organize and present commands in a structured manner, allowing for better user interaction and navigation.

Properties

 Expand table

Property	Type	Description
Commands	IContextItem[]	The collection of commands that will be displayed on the page.
Details	IDetails	The details associated with the page. This can include additional information or context about the commands displayed.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<object, IItemsChangedEventArgs > ItemsChanged	Occurs when the items in the page have changed. This event can be used to update the UI or perform other actions when the commands or details are modified.

Methods

Method	Description
GetContent()	Retrieves the content of the page, including the commands and details. This method can be used to access the current state of the page and its items.
RaiseItemsChanged(Int)	Raises the ItemsChanged event with the new number of items following the change. This method can be used to notify subscribers about changes to the items displayed on the page.

ContentPage.GetContent() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **GetContent** method retrieves the content of the page, including the commands and details. This method can be used to access the current state of the page and its items.

Returns

An [IContent\[\]](#) array containing the content of the page. This array includes all the items that are currently displayed on the page, allowing you to access their properties and perform actions on them.

ContentPage.RaiseItemsChanged(Int) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Raises the **ItemsChanged** event with the new total number of items. This method can be used to notify subscribers about changes in the items displayed on the page.

Parameters

totalItems Int

The new total number of items in the collection. This parameter is used to indicate the current number of items following a change.

CopyTextCommand Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [InvokableCommand](#)

The **CopyTextCommand** class is used to define a command that copies text to the clipboard. It is a specialized command that provides functionality for copying text and displaying a result message in the Command Palette.

Constructors

[Expand table](#)

Constructor	Description
CopyTextCommand(String)	Initializes the command with a text parameter, sets its name to "Copy", and adds an icon.

Properties

[Expand table](#)

Property	Type	Description
Result	CommandResult	What happens in the palette after the command is executed. Defaults to CommandResult.ShowToast("Copied to clipboard") .
Text	String	Gets and sets the text of the command.

Methods

[Expand table](#)

Method	Description
Invoke()	Sets the clipboard text to the value of Text and returns the Result .

CopyTextCommand(String) Constructor

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [CopyTextCommand](#) class with the command text set to the *text* parameter, its name set to "Copy", and an icon added.

C#

```
public CopyTextCommand(string text)
{
    Text = text;
    Name = "Copy";
    Icon = new IconInfo("\uE8C8");
}
```

Parameters

text String

The text to be copied to the clipboard. This parameter is used to set the **Text** property of the command.

CopyTextCommand.Invoke() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Sets the clipboard text to the value of [Text](#) and returns the [Result](#).

Returns

An [ICommandResult](#) object that represents the result of the command execution.

Details Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [IDetails](#)

The **Details** class is used to define the details section of an item in the Command Palette. It provides properties for setting the title, body, and metadata of the details section.

Properties

 [Expand table](#)

Property	Type	Description
Body	String	The body of the details section. This is the main content that will be displayed in the details section.
HeroImage	IconInfo	The hero image associated with the details section. This is an image that represents the details section visually.
Metadata	IDetailsElement	The metadata associated with the details section. This is additional information that can be displayed in the details section.
Title	String	The title of the details section. This is the main heading that will be displayed in the details section.

DetailsCommand Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IDetailsCommand](#)

The **DetailsCommand** class is used to define a command that can be executed in the Command Palette. It is a specialized command that provides additional details about the command's execution.

Properties

 Expand table

Property	Type	Description
Command	ICommand	The command associated with the details command. This is the command that will be executed when the details command is invoked.

DetailsElement Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IDetailsElement](#)

The **DetailsElement** class is used to define a details element that can be displayed in the details section of an item in the Command Palette.

Properties

 Expand table

Property	Type	Description
Key	String	The key of the details element. This is used to identify the element in the Command Palette.
Data	IDetailsData	The data associated with the details element.

DetailsLink Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IDetailsLink](#)


The **DetailsLink** class is used to define a link that can be displayed in the details section of an item in the Command Palette. This class is useful for providing additional information or resources related to the item, allowing users to easily access relevant content.

Constructors

 Expand table

Constructor	Description
DetailsLink()	Initializes a new instance of the DetailsLink class.
DetailsLink(String)	Initializes a new instance of the DetailsLink class with the specified text.
DetailsLink(String, String)	Initializes a new instance of the DetailsLink class with the specified text and link.

Properties

 Expand table

Property	Type	Description
Link	Uri	The URI of the link.
Text	String	The text to display for the link.

DetailsLink Constructors

Article • 03/31/2025

DetailsLink() Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [DetailsLink](#) class.

C#

```
public DetailsLink()
{
}
```

DetailsLink(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [DetailsLink](#) class, setting its [Link](#) and [Text](#) properties to *url*.

C#

```
public DetailsLink(string url)
    : this(url, url)
{
}
```

Parameters

url String

The URL to be set as both the link and text for the [DetailsLink](#) instance.

DetailsLink(String, String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [DetailsLink](#) class, setting its [Link](#) property to *url* and [Text](#) to *text*.

C#

```
public DetailsLink(string url, string text)
{
    if (Uri.TryCreate(url, default(UriCreationOptions), out var newUri))
    {
        Link = newUri;
    }

    Text = text;
}
```

Parameters

url String

The URL to be set as the link for the [DetailsLink](#) instance.

text String

The text to be displayed for the link in the Command Palette. This text is what users will see and click on to access the link.

DetailsSeparator Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IDetailsSeparator](#)

The **DetailsSeparator** class is used to define a separator that can be displayed in the details section of an item in the Command Palette. This class is useful for visually separating different sections of information, making it easier for users to read and understand the details associated with an item.

DetailsTags Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IDetailsTags](#)

The **DetailsTags** class is used to define a set of tags that can be associated with items in the Command Palette. Tags are useful for categorizing or labeling items, making it easier for users to find and filter them based on specific criteria.

Properties

 Expand table

Property	Type	Description
Tags	ITag[]	An array of tags associated with the item.

DynamicListPage Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [ListPage](#)

Implements [IDynamicListPage](#)


The **DynamicListPage** class is a specialized version of the [ListPage](#) class that allows for dynamic updates to the list of items displayed in the command palette. This class is useful for scenarios where the list of items may change frequently or needs to be updated based on user interactions or other events.

Properties

 Expand table

Property	Type	Description
SearchText	String	Overrides the SearchText property of the ListPage class.

Methods

 Expand table

Method	Description
UpdateSearchText(String, String)	Updates the search text for the dynamic list page.

DynamicListPage.UpdateSearchText(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **UpdateSearchText** method is used to update the search text for the dynamic list page. This allows the extension to modify the search criteria dynamically based on user interactions or other events.

Parameters

oldSearch String

The previous search text that was used for filtering the list items. This parameter is useful for determining what has changed in the search criteria.

newSearch String

The new search text that will be used for filtering the list items. This parameter represents the updated search criteria that should be applied to the list of items displayed in the command palette.

ExtensionHost Class


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)


The **ExtensionHost** class is a base class for creating extension hosts in the Command Palette. It provides methods for logging messages, showing and hiding status messages, and initializing the extension host with an instance of [IExtensionHost](#).

Properties

 Expand table

Property	Type	Description
Host	IExtensionHost	Gets the instance of the extension host that is associated with this extension. This property is used to access the extension host's methods and properties.

Methods

 Expand table

Method	Description
HideStatus(IStatusMessage)	Hides the status message that is currently being displayed in the command palette.
Initialize(IExtensionHost)	Initializes the extension host with an instance of IExtensionHost .
LogMessage(ILogMessage)	Logs a message to the command palette.
LogMessage(String)	Logs a message to the command palette.
ShowStatus(IStatusMessage)	Shows a status message in the command palette.

ExtensionHost.HideStatus([IStatusMessage](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **HideStatus** method is used to hide a status message that was previously displayed in the command palette. This is useful for clearing the status area when the operation associated with the status message is complete or no longer relevant.

Parameters

message [IStatusMessage](#)

The status message to be hidden, provided as an object implementing **IStatusMessage**. This parameter identifies the specific status message that should be removed from the command palette's status area.

ExtensionHost.Initialize([IExtensionHost](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Initialize** method is called when the extension host is initialized. This method is typically used to set up any necessary resources or configurations for the extension.

Parameters

host [IExtensionHost](#)

The extension host instance that is being initialized. This parameter provides access to the extension host's functionality and services, allowing the extension to interact with the command palette and other components of the system.

ExtensionHost.LogMessage([ILogMessage](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LogMessage** method logs a message to the command palette's output window. This is useful for debugging and providing feedback to the user about the extension's operations.

Parameters

message [ILogMessage](#)

The message to be logged, provided as an object implementing **ILogMessage**. This message can contain any relevant information that the developer wants to provide to the user or for debugging purposes.

ExtensionHost.LogMessage(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LogMessage** method logs a message to the command palette's output window. This is useful for debugging and providing feedback to the user about the extension's operations.

Parameters

message String

The message to be logged, provided as a string. This message can contain any relevant information that the developer wants to provide to the user or for debugging purposes.

ExtensionHost.ShowStatus([IStatusMessage](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ShowStatus** method displays a status message in the command palette. This method is typically used to provide feedback to the user about the current state of the extension or to indicate that a long-running operation is in progress.

Parameters

message [IStatusMessage](#)

FallbackCommandItem Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [CommandItem](#)

Implements [IFallbackCommandItem](#), [IFallbackHandler](#)


The **FallbackCommandItem** class is used to create a command item that serves as a fallback option in the command palette. It allows you to define a command that will be executed when no other commands match the user's input.

Constructors

 Expand table


Constructor	Description
FallbackCommandItem(ICommand)	Initializes a new instance of the FallbackCommandItem class with the specified command.

Properties

 Expand table

Property	Type	Description
FallbackHandler	IFallbackHandler	Gets or sets the fallback handler that will be used to execute the command.

Methods

 Expand table

Method	Description
UpdateQuery(String)	Updates the query string for the command item.

FallbackCommandItem Constructors

Article • 03/31/2025

FallbackCommandItem(ICommand) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [FallbackCommandItem](#) class with the provided *command* and sets its [_fallbackHandler](#) property if *command* implements [IFallbackHandler](#).

C#

```
public FallbackCommandItem(ICommand command)
    : base(command)
{
    if (command is IFallbackHandler f)
    {
        _fallbackHandler = f;
    }
}
```

Parameters

command [ICommand](#)

The command that will be used to create the fallback command item. The *command* parameter must implement the [ICommand](#) interface.

FallbackCommandItem.UpdateQuery(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **UpdateQuery** method updates the query string for the command item. This method is used to modify the query that is used to filter the command items in the command palette.

Parameters

query String

The query string that will be used to filter the command items. The *query* parameter is used to match against the command item's name and other properties to determine if it should be displayed in the command palette.

Filter Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IFilter](#)

The **Filter** class is used to filter a list of items based on a specified criteria. It provides methods to apply the filter and retrieve the filtered results.

Properties

 Expand table

Property	Type	Description
Icon	IIconInfo	Not yet implemented.
Id	String	Not yet implemented.
Name	String	Not yet implemented.

FormContent Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [IFormContent](#)

The **FormContent** class is used to create a form within the Command Palette. It provides properties and methods to manage the form's data, state, and template.

Properties

 Expand table

Property	Type	Description
DataJson	String	The JSON representation of the form's data.
StateJson	String	The JSON representation of the form's state.
TemplateJson	String	The JSON representation of the form's template.

Methods

 Expand table

Method	Description
SubmitForm(String)	Submits the form with the specified JSON data.
SubmitForm(String, String)	Submits the form with the specified JSON data and state.

FormContent.SubmitForm(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SubmitForm** method submits the form with the specified JSON data.

Parameters

inputs **String**

The JSON representation of the form's inputs.

Returns

An [ICommandResult](#) object that contains the result of the form submission.

FormContent.SubmitForm(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SubmitForm** method submits the form with the specified JSON data and state.

Parameters

inputs **String**

The JSON representation of the form's inputs.

data **String**

The JSON representation of the form's state data.

Returns

An [ICommandResult](#) object that contains the result of the form submission.

GoToPageArgs Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IGoToPageArgs](#)

The **GoToPageArgs** class is used to represent the arguments for navigating to a page in the Command Palette. It provides properties for the type of navigation mode and the ID of the page to navigate to.

Properties

 Expand table

Property	Type	Description
NavigationMode	NavigationMode	Gets or sets the navigation mode. Defaults to NavigationMode.Push .
PageId	String	Required. Gets or sets the ID of the page.

IconData Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IIconData](#)


The **IconData** class is used to represent icon data for a command in the Command Palette. It provides properties to access the icon data and its associated URI.

Constructors

 Expand table

Constructor	Description
IconData(IRandomAccessStreamReference)	Initializes the Data property with the provided IRandomAccessStreamReference .
IconData(String)	Initializes the Icon property with the provided String .

Properties

 Expand table

Property	Type	Description
Data	Windows.Storage.Streams.IRandomAccessStreamReference	Gets or sets the icon data as a stream reference. This property is used to load the icon data from a stream.
Icon	String	Gets or sets the icon URI. This property is used to load the icon from a URI.

IconData Constructors

Article • 03/31/2025

IconData(IRandomAccessStreamReference) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [IconData](#) class with the [Data](#) property set to *data*.

C#

```
public IconData(IRandomAccessStreamReference data)
{
    Data = data;
}
```

Parameters

data [Windows.Storage.Streams.IRandomAccessStreamReference](#)

The icon data as a stream reference. This property is used to load the icon data from a stream.

IconData(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [IconData](#) class with the [Icon](#) property set to *icon*.

C#

```
public IconData(string? icon)
{
    Icon = icon;
}
```

Parameters

icon String

The icon URI. This property is used to load the icon from a URI.

IconHelpers Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Includes helper methods for creating [IconData](#) objects from relative paths. This is useful for loading icons from specific locations in your project or application.

Methods

 Expand table

Method	Description
FromRelativePath(String)	
FromRelativePaths(String, String)	

IconHelpers.FromRelativePath(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

This method creates an [IconInfo](#) object from a relative path. This is useful for loading icons that have different appearances based on the current theme.

Parameters

path String

The relative path to the icon. This path should point to an image file can be used for both light and dark themes.

Returns

An [IconInfo](#) object that contains the light and dark mode icons.

IconHelpers.FromRelativePaths(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

This method creates an [IconInfo](#) object from two relative paths, one for light mode and one for dark mode. This is useful for loading icons that have different appearances based on the current theme.

Parameters

lightIcon **String**

The relative path to the icon for light mode. This path should point to an image file that represents the icon in a light theme.

darkIcon **String**

The relative path to the icon for dark mode. This path should point to an image file that represents the icon in a dark theme.

Returns

An [IconInfo](#) object that contains the light and dark mode icons.

IconInfo Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IIconInfo](#)

The **IconInfo** class is used to define icons for commands in the Command Palette. It allows you to specify different icons for light and dark modes, ensuring that your icons are visually appealing and consistent with the user's theme preferences.

Constructors

 Expand table

Constructor	Description
IconInfo(IconData, IconData)	Initializes the icon with a dark mode and a light mode version.
IconInfo(String)	Initializes the icon with one version of the icon, used for both light and dark modes.

Properties

 Expand table

Property	Type	Description
Dark	IconData	Gets or sets the dark mode version of the icon.
Light	IconData	Gets or sets the light mode version of the icon.

IconInfo Constructors

Article • 03/31/2025

IconInfo(IconData, IconData) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [IconInfo](#) class with a dark mode and a light mode version of the icon.

C#

```
public IconInfo(IconData light, IconData dark)
{
    Light = light;
    Dark = dark;
}
```

Parameters

dark [IconData](#)

The dark mode version of the icon. This parameter is used to specify the icon that will be displayed when the application is in dark mode.

light [IconData](#)

The light mode version of the icon. This parameter is used to specify the icon that will be displayed when the application is in light mode.

IconInfo(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [IconInfo](#) class with one version of the icon, used for both light and dark modes.

C#

```
public IconInfo(string icon)
{
    Dark = Light = new(icon);
}
```

Parameters

icon String

The icon that will be displayed in both light and dark modes. This parameter is used to specify a single icon that will be used regardless of the application's theme.

InvokableCommand Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Command](#)

Implements [IInvokableCommand](#)

The **InvokableCommand** class is a specialized command that can be invoked directly from the command palette. It provides a way to execute commands without requiring additional user input or confirmation.

Methods

 [Expand table](#)

Method	Description
Invoke()	Keeps the command palette open after the command is invoked.
Invoke(Object)	Keeps the command palette open after the command is invoked with an object parameter.

InvokableCommand.Invoke Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Invoke** method is used to execute the command associated with the [InvokableCommand](#) instance. This method is typically called when the command is triggered by the user. Keeps the command palette open after the command is invoked.

Returns

A [CommandResult.KeepOpen\(\)](#) object that indicates the command palette should remain open after the command is executed. This allows the user to continue interacting with the command palette without having to reopen it.

InvokableCommand.Invoke(Object) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

This method executes the command associated with the [InvokableCommand](#) instance and keeps the command palette open.

Parameters

sender Object

The object that is invoking the command. This parameter can be used to pass additional context or data to the command being executed. It can be any object type, depending on the specific implementation of the command.

Returns

An [InvokableCommand.Invoke\(\)](#).

ISettingsForm Interface

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ISettingsForm** interface is used to represent a form in the Command Palette settings. It provides methods for converting the form to different representations, such as a data identifier, dictionary, or state.

Methods

 Expand table

Method	Description
ToDataIdentifier()	Converts the settings form to a data identifier.
ToDictionary()	Converts the settings form to a dictionary representation.
ToForm()	Converts the settings form to a form representation.
ToState()	Converts the settings form to a state representation.
Update(JsonObject)	Updates the settings form with the specified JSON object.

ISettingsForm.ToDataIdentifier() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDataIdentifier** method returns a string that represents the data identifier of the settings form. This identifier is used to uniquely identify the settings form in the context of the command palette extension.

Returns

A **String** that contains the data identifier of the settings form.

ISettingsForm.ToDictionary() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDictionary** method returns a dictionary containing a representation of the settings form's data.

Returns

A **Dictionary<String, Object>** that contains the settings form's data. The keys in the dictionary are the names of the settings, and the values are the corresponding values of those settings.

ISettingsForm.ToForm() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToForm** method converts the settings form to a form representation.

Returns

A **String** that represents the settings as a form.

ISettingsForm.ToState() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToState** method converts the settings form to a state representation. This method is used to obtain a string representation of the settings form's state.

Returns

A **String** that represents the state of the settings form.

ISettingsForm.Update(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the settings form with the specified JSON object. This method is used to apply changes to the settings form based on the provided JSON data.

Parameters

payload **JsonObject**

The **JsonObject** containing the data to update the settings form. This parameter is required and cannot be null.

ItemsChangedEventArgs Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IItemsChangedEventArgs](#)


The **ItemsChangedEventArgs** class is used to represent the event arguments for the items changed event in the Command Palette Extensions Toolkit. It contains information about the total number of items that have changed.

Constructors

 Expand table

Constructor	Description
ItemsChangedEventArgs(Integer)	Initializes a new instance of the ItemsChangedEventArgs class with the specified total number of items.

Properties

 Expand table

Property	Type	Description
TotalItems	Integer	Gets the total number of items that have changed. This property is read-only.

ItemsChangedEventArgs Constructors

Article • 03/31/2025

ItemsChangedEventArgs(Integer) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ItemsChangedEventArgs](#) class with the [TotalItems](#) property set to *totalItems* that defaults to `-1`.

C#

```
public ItemsChangedEventArgs(int totalItems = -1)
{
    TotalItems = totalItems;
}
```

Parameters

totalItems Integer

The total number of items that have changed. This parameter is optional and defaults to `-1` if not provided.

JsonSettingsManager Class


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **JsonSettingsManager** class provides functionality for managing settings in JSON format. It allows loading and saving settings to a specified file path.

Properties

 Expand table

Property	Type	Description
FilePath	String	The file path where the settings are stored.
Settings	Settings	The settings object that contains the configuration data.

Methods

 Expand table

Method	Description
LoadSettings()	Loads settings from the specified file path.
SaveSettings()	Saves the current settings to the specified file path.

JsonSettingsManager.LoadSettings Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LoadSettings** method loads the settings from the specified file path in JSON format. It deserializes the JSON data into the settings object.

JsonSettingsManager.SaveSettings Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SaveSettings** method saves the current settings to the specified file path in JSON format.

KeyChordHelpers Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

This class provides helper methods for working with key chords in the Command Palette.

Methods

 Expand table

Method	Description
FromModifiers (Boolean, Boolean, Boolean, Boolean, Integer, Integer)	Creates a key chord from the specified modifier keys and key codes.

KeyChordHelpers.FromModifiers(Boolean, Boolean, Boolean, Boolean, Integer, Integer) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FromModifiers** method creates a new [KeyChord](#) object from the specified modifier keys and virtual key code.

Parameters

ctrl **Boolean**

Indicates whether the `Control` key is pressed.

alt **Boolean**

The `Alt` key is pressed.

shift **Boolean**

The `Shift` key is pressed.

win **Boolean**

The `Windows` key is pressed.

vkey **Integer**

The virtual key code of the key that is pressed. This value is typically obtained from the `KeyInterop` class.

scanCode **Integer**

The scan code of the key that is pressed. This value is typically obtained from the `KeyInterop` class.

Returns

A [KeyChord](#) object that represents the specified modifier keys and virtual key code.

ListHelpers Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ListHelpers** class provides static methods for working with lists of items in the command palette. It includes methods for filtering, scoring, and updating lists of items.

Methods

 Expand table

Method	Description
FilterList(IEnumerable<IListItem>, String)	Filters a list of items based on a search string.
FilterList<T>(IEnumerable<T>, String, Func<String, T, Integer>)	Filters a list of items based on a search string and a scoring function.
InPlaceUpdateList<T>(IList<T>, IEnumerable<T>)	Updates a list of items in place.
ScoreListItem(String, ICommandItem)	Scores a list item based on a search string and a command item.

Structs

 Expand table

Struct	Description
ScoredListItem	Represents a scored list item.
Scored<T>	Represents a scored item of type T.

ListHelpers.FilterList(IEnumerable<IListItem>, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FilterList** method filters a list of items based on a query string. It returns a new list containing only the items that match the query.

Parameters

items [IEnumerable<IListItem>](#)

The list of items to filter.

query **String**

The query string used to filter the items. The filtering is case-insensitive and checks if the query is contained within the item's name or description.

Returns

An [IEnumerable<IListItem>](#) containing the filtered items that match the query.

ListHelpers.FilterList<T> (IEnumerable<T>, String, Func<String, T, Integer>) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FilterList** method filters a list of items based on a query string and a scoring function. It returns a new list containing only the items that match the query, sorted by their score.

Parameters

items **IEnumerable<T>**

The list of items to filter.

query **String**

The query string used to filter the items. The filtering is case-insensitive and checks if the query is contained within the item's name or description.

scoreFunction **Func<String, T, Integer>**

The function used to calculate the score of each item based on the query. The score is used to sort the filtered items.

Returns

An **IEnumerable<T>** containing the filtered items that match the query, sorted by their score in descending order. The items with a higher score appear first in the list.

ListHelpers.InPlaceUpdateList<T> (IList<T>, IEnumerable<T>) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **InPlaceUpdateList<T>** method updates the contents of an existing list in place with new contents. This is useful for modifying a list without creating a new instance, which can be more efficient in terms of memory and performance.

Parameters

original **IList<T>**

The original list to be updated. This list will be modified in place.

newContents **IEnumerable<T>**

The new contents to be added to the original list. This can be any collection of items that implement the **IEnumerable<T>** interface.

ListHelpers.ScoreListItem(String, ICommandItem) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ScoreListItem** method is a static method that scores a list item based on a search string and a command item. It is used to determine how well the list item matches the search criteria.

Parameters

query String

The search string used to score the list item. This string is typically the text entered by the user in the command palette.

listItem [ICommandItem](#)

The command item to be scored. This item represents a command or action that can be executed in the command palette.

Returns

An **Integer** representing the score of the list item based on the search string and command item.

ListHelpers.ScoredListItem Struct

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ScoredListItem** struct represents a scored list item. It's used to encapsulate a list item along with its score, which indicates how well the item matches a search query.

Fields

 Expand table

Field	Type	Description
ListItem	IListItem	The list item being scored.
Score	Integer	The score of the list item. A higher score indicates a better match to the search query.

ListHelpers.Scored<T> Struct

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Scored<T>** struct represents a scored item of type **T**. It's used to encapsulate an item along with its score, which indicates how well the item matches a search query.

Fields

 Expand table

Field	Type	Description
Item	T	The item being scored.
Score	Integer	The score of the item. A higher score indicates a better match to the search query.

ListItem Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [CommandItem](#)

Implements [IListItem](#)

The **ListItem** class represents an item in a list. It is used to display items in the command palette and provides properties and methods for managing the item's appearance and behavior.

Constructors

 Expand table

Constructor	Description
ListItem(ICommand)	Initializes a new instance of the ListItem class with the specified command.
ListItem(ICommandItem)	Initializes a new instance of the ListItem class with the specified command item.

Properties

 Expand table

Property	Type	Description
Details	IDetails	Provides additional details about the item.
Section	String	The section to which the item belongs. This property is used to group items in the list.
Tags	ITag[]	The tags associated with the item. Tags are used to categorize items and can be used for filtering.
TextToSuggest	String	The text to suggest for the item. This property is used to provide suggestions for the item when the user types in the command palette.

ListItem Constructors

Article • 03/31/2025

ListItem([ICommand](#)) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ListItem](#) class with an [ICommand](#) instance.

C#

```
public ListItem(ICommand command)
    : base(command)
{
}
```

Parameters

command [ICommand](#)

The command associated with the list item. This command is executed when the item is selected in the command palette.

ListItem([ICommandItem](#)) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ListItem](#) class with an [ICommandItem](#) instance.

C#

```
public ListItem(ICommandItem command)
    : base(command)
{
}
```

Parameters

command [ICommandItem](#)

The command item associated with the list item. This command item is executed when the item is selected in the command palette.

ListPage Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [Page](#), [IListPage](#)


The **ListPage** class defines a page that displays a list of items. It provides properties and methods to manage the display and interaction with the list of items.

Properties

 Expand table


Property	Type	Description
EmptyContent	ICommandItem	The content to display when the list is empty.
Filters	IFilters	The filters to apply to the list of items.
GridProperties	IGridProperties	The properties for the grid layout of the list.
HasMoreItems	Boolean	Indicates if there are more items to load.
PlaceholderText	String	The placeholder for the filter on the page.
SearchText	String	The text to filter the list of items.
ShowDetails	Boolean	Indicates if the details of the items should be shown.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<object, IItemsChangedEventArgs > ItemsChanged	Occurs when the items in the list have changed.

Methods

 Expand table

Method	Description
GetItems()	Returns the list of items.
LoadMore()	Loads more items into the list.
RaiseItemsChanged(Integer)	Raises the ItemsChanged event.

ListItems.GetPage() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **GetPage** method retrieves the items in the current page of the list. This method is typically called when the user navigates to a different page in the list.

Returns

An [IListItem\[\]](#) array containing the items in the current page of the list. The number of items returned is determined by the **PageSize** property of the **ListItems** class.

ListItems.LoadMore() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LoadMore** method is used to load more items into the list. This method is typically called when the user scrolls to the end of the list and more items need to be loaded.

ListItems.RaiseItemsChanged(Integer) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **RaiseItemsChanged** method raises the **ItemsChanged** event, indicating that the items in the list have changed.

Parameters

totalItems Integer

The total number of items in the list after the change.

LogMessage Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [ILogMessage](#)


The **LogMessage** class is used to define a log message in the command palette. It provides properties to specify the message content and its state.

Constructors

 Expand table

Constructor	Description
LogMessage(String)	Initializes a new instance of the LogMessage class with the specified message.

Properties

 Expand table

Property	Type	Description
Message	String	The content of the log message.
State	MessageState	The state of the log message.

LogMessage Constructors

Article • 03/31/2025

LogMessage(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [LogMessage](#) class with a *message*, which defaults to an empty string if no argument is provided.

C#

```
public LogMessage(string message = "")
{
    _message = message;
}
```

Parameters

message **String**

The content of the log message. This parameter is optional and defaults to an empty string if not provided.

MarkdownContent Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [IMarkdownContent](#)

The **MarkdownContent** class is used to represent the content of a command palette item in markdown format. It provides properties to specify the body of the markdown content and whether it should be rendered as HTML.

Properties

 [Expand table](#)

Property	Type	Description
Body	String	The body of the markdown content. This property is required and cannot be null or empty.

MatchOption Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **MatchOption** class is used to define the options for matching in the command palette. It provides properties to specify how the matching should be performed, including case sensitivity and prefix/suffix matching.

Properties

 Expand table

Property	Type	Description
IgnoreCase	Boolean	Indicates whether the match should ignore case sensitivity.
Prefix	String	Indicates whether the match should be a prefix match.
Suffix	String	Indicates whether the match should be a suffix match.

MatchResult Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **MatchResult** class is used to represent the result of a match operation in the Command Palette Extensions Toolkit. It contains information about whether the match was successful, the score of the match, and any additional data related to the match.

Constructors

 Expand table


Constructor	Description
MatchResult(Boolean, SearchPrecisionScore)	Initializes a new instance of the MatchResult class with a success flag and a search precision score.
MatchResult(Boolean, SearchPrecisionScore, List<Integer>, Integer)	Initializes a new instance of the MatchResult class with a success flag, a search precision score, match data, and a raw score.

Properties

 Expand table

Property	Type	Description
MatchData	List<Integer>	A list of integers representing the match data.
RawScore	Integer	The raw score of the match.
Score	Integer	The score of the match.
SearchPrecision	SearchPrecisionScore	The precision score of the match.
Success	Boolean	A boolean value indicating whether the match was successful.

Methods

 Expand table

Method	Description
IsSearchPrecisionScoreMet()	Checks if the search precision score is met.

MatchResult Constructors

Article • 03/31/2025

MatchResult(Boolean, [SearchPrecisionScore](#)) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [MatchResult](#) class with its [Success](#) property set to *success* and [SearchPrecision](#) set to *searchPrecision*.

C#

```
public MatchResult(bool success, SearchPrecisionScore searchPrecision)
{
    Success = success;
    SearchPrecision = searchPrecision;
}
```

Parameters

success **Boolean**

Indicates whether the match was successful.

searchPrecision [SearchPrecisionScore](#)

The search precision score for the match. This score is used to determine how closely the match aligns with the search criteria.

MatchResult(Boolean, [SearchPrecisionScore](#), List<Integer>, Integer) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [MatchResult](#) class with its [Success](#) property set to *success*, [SearchPrecision](#) set to *searchPrecision*, [MatchData](#) set to *matchData*, and [RawScore](#) set to *rawScore*.

C#

```
public MatchResult(bool success, SearchPrecisionScore searchPrecision,
    List<int> matchData, int rawScore)
{
    Success = success;
    SearchPrecision = searchPrecision;
    MatchData = matchData;
    RawScore = rawScore;
}
```

Parameters

success **Boolean**

Indicates whether the match was successful.

searchPrecision [SearchPrecisionScore](#)

The search precision score for the match. This score is used to determine how closely the match aligns with the search criteria.

matchData **List<Integer>**

The list of match data. This data can be used to provide additional context or information about the match.

rawScore **Integer**

The raw score for the match. This score is used to quantify the quality of the match based on the search criteria.

MatchResult.IsSearchPrecisionScoreMet() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **IsSearchPrecisionScoreMet** method checks if the search precision score is met.

Returns

A **Boolean** value indicating whether the search precision score is met.

NoOpCommand Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [InvokableCommand](#)

The **NoOpCommand** class is used to represent a command that does nothing when invoked. It is typically used as a placeholder or for testing purposes.

Methods

 Expand table

Method	Description
Invoke()	Invokes the command. This method does nothing and is used as a placeholder.

NoOpCommand.Invoke() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **NoOpCommand.Invoke()** method is used to invoke the command. This method does nothing and is used as a placeholder.

Returns

An [ICommandResult](#) object that indicates the result of the command invocation.

OpenUrlCommand Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [InvokableCommand](#)

The **OpenUrlCommand** class is used to represent a command that opens a URL in the default web browser.

Constructors

 Expand table

Constructor	Description
OpenUrlCommand(String)	Initializes a new instance of the OpenUrlCommand class with the specified URL.

Properties

 Expand table

Property	Type	Description
Result	CommandResult	Gets the result of the command. This property is used to indicate whether the command was successful or not.

Methods

 Expand table

Method	Description
Invoke()	Invokes the command. This method opens the specified URL in the default web browser.

OpenUrlCommand Constructors

Article • 03/31/2025

OpenUrlCommand(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [OpenUrlCommand](#) class with the URL target set to *target*, its name set to `Open`, and an icon added.

C#

```
public OpenUrlCommand(string target)
{
    _target = target;
    Name = "Open";
    Icon = new IconInfo("\uE8A7");
}
```

Parameters

target **String**

The URL to be opened by the command.

OpenUrlCommand.Invoke() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Invokes the command. This method opens the specified URL in the default web browser.

Returns

A [CommandResult](#) object that indicates the result of the command invocation.

Page Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Command](#)

Implements [IPage](#)

Represents an additional "nested" page within the Command Palette.

Properties

 Expand table

Property	Type	Description
AccentColor	OptionalColor	The accent color of the page. This property is used to set the color of the page header and other UI elements.
IsLoading	Boolean	Gets or sets a value indicating whether the page is loading. This property is used to show a loading indicator while the page is being loaded.
Title	String	The title of the page. This property is used to set the text displayed in the page header.

ProgressState Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [IProgressState](#)

The **ProgressState** class is used to represent the state of a progress indicator. It provides properties to indicate whether the progress is indeterminate and to set the progress percentage.

Properties

 Expand table

Property	Type	Description
IsIndeterminate	Boolean	Gets or sets a value indicating whether the progress is indeterminate.
ProgressPercent	UInt	Gets or sets the progress percentage. This value is between 0 and 100.

PropChangedEventArgs Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IPropChangedEventArgs](#)


The **PropChangedEventArgs** class is used to provide data for the property changed event. It contains information about the property that has changed.

Constructors

 Expand table

Constructor	Description
PropChangedEventArgs(String)	Initializes a new instance of the PropChangedEventArgs class with the specified property name.

Properties

 Expand table

Property	Type	Description
PropertyName	String	Gets the name of the property that has changed.

PropChangedEventArgs Constructors

Article • 03/31/2025

PropChangedEventArgs(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [PropChangedEventArgs](#) class with the [PropertyName](#) property set to *propertyName*.

C#

```
public PropChangedEventArgs(string propertyName)
{
    PropertyName = propertyName;
}
```

Parameters

propertyName **String**

The name of the property that has changed. This value is used to set the [PropertyName](#) property of the [PropChangedEventArgs](#) class.

SearchPrecisionScore Enum


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SearchPrecisionScore** enum is used to specify the precision score of a search. The precision score is a measure of how closely the search results match the user's query. A higher precision score indicates that the search results are more relevant to the user's query.

Fields

 Expand table

Name	Value	Description
None	0	No precision score.
Low	20	Considered a low precision score.
Regular	50	Considered a regular precision score.

Setting<T> Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [ISettingsForm](#)


The **Setting<T>** class represents a setting in the command palette extension. It contains properties and methods to define and manage settings, including their keys, values, labels, descriptions, and error messages. This class is used to create settings that can be accessed and modified by the command palette extension.

Constructors

 Expand table

Constructor	Description
Setting()	The default constructor for the Setting class.
Setting(String, T)	The constructor for the Setting class that takes a key and a value.
Setting(String, String, String, T)	The constructor for the Setting class that takes a key, label, description, and value.

Properties

 Expand table

Property	Type	Description
Description	String	Gets or sets the description of the setting.
ErrorMessage	String	Gets or sets the text of the error message.
IsRequired	Boolean	Gets or sets whether the setting is required.
Key	String	Gets the key of the setting.
Label	String	Gets or sets the label of the setting.

Property	Type	Description
Value	T	Gets or sets the value of the setting.

Methods

 Expand table

Method	Description
ToDataIdentifier()	The ToDataIdentifier method converts the setting to a data identifier.
ToDictionary()	The ToDictionary method converts the setting to a dictionary.
ToForm()	The ToForm method converts the setting to a form.
ToState()	The ToState method converts the setting to a state.
Update(JsonObject)	The Update method updates the setting with the specified JSON object.

Setting Constructors

Article • 03/31/2025

Setting() Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [Setting](#) class with [Value](#) property set to its default value and [Key](#) set to an empty string.

C#

```
protected Setting()
{
    Value = default;
    Key = string.Empty;
}
```

Setting(String, T) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [Setting](#) class with its [Key](#) property set to *key* and its [Value](#) set to *defaultValue*.

C#

```
public Setting(string key, T defaultValue)
{
    Key = key;
    Value = defaultValue;
}
```

Parameters

key String

The key of the setting. This is a unique identifier for the setting.

defaultValue T

The default value of the setting. This is the value that will be used if no value is provided for the setting.

Setting(String, String, String, T) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [Setting](#) class with its [Key](#) property set to *key*, its [Value](#) set to *defaultValue*, its [Label](#) set to *label*, and its [Description](#) set to *description*.

C#

```
public Setting(string key, string label, string description, T defaultValue)
{
    Key = key;
    Value = defaultValue;
    Label = label;
    Description = description;
}
```

Parameters

key String

The key of the setting. This is a unique identifier for the setting.

label String

The label of the setting. This is a user-friendly name for the setting that will be displayed in the UI.

description String

The description of the setting. This is a brief explanation of what the setting does and how it should be used.

defaultValue T

The default value of the setting. This is the value that will be used if no value is provided for the setting.

Setting.ToDataIdentifier() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDataIdentifier** method converts the setting to a data identifier. This method is used to create a data identifier object that represents the current value of the setting. The data identifier object can be used to store the current value of the setting in a format that can be easily serialized and deserialized.

Returns

A **String** that represents the current value of the setting.

Setting.ToDictionary() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDictionary** method converts the setting to a dictionary. This method is used to create a dictionary object that represents the current value of the setting. The dictionary object can be used to store the current value of the setting in a format that can be easily serialized and deserialized.

Returns

A **Dictionary<String, Object>** that represents the current value of the setting. The dictionary contains the key-value pairs that represent the current value of the setting.

Setting.ToForm() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToForm** method converts the setting to a form. This method is used to create a form object that represents the current value of the setting. The form object can be used to store the current value of the setting in a format that can be easily serialized and deserialized.

Returns

A **String** that represents the current value of the setting.

Setting.ToState() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToState** method converts the setting to a state. This method is used to create a state object that represents the current value of the setting. The state object can be used to store the current value of the setting in a format that can be easily serialized and deserialized.

Returns

A **String** that represents the current value of the setting.

Setting.Update(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the setting with the specified *payload* parameter. This method is used to modify the value of the setting based on the provided **JsonObject**. It must contain a property with the same name as the setting's key, and the value of that property must be of the same type as the setting's value.

Parameters

payload **JsonObject**

The **JsonObject** that contains the new value for the setting. The **JsonObject** must contain a property with the same name as the setting's key, and the value of that property must be of the same type as the setting's value.

Settings Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [ICommandSettings](#)


The **Settings** class is used to manage the settings of a command palette extension. It provides methods to add, retrieve, and update settings, as well as to convert settings to JSON format.

Properties

 Expand table


Property	Type	Description
SettingsPage	IContentPage	The settings page associated with the command palette extension.

Events

 Expand table

Event	Description
Windows.Foundation.TypedEventHandler<object, Settings> SettingsChanged	Occurs when the settings of the command palette extension are changed.

Methods

 Expand table

Method	Description
Add<T>(Setting<T>)	Adds a new setting to the command palette extension.
GetSetting<T>(String)	Retrieves the value of a setting by its name.

Method	Description
ToContent()	Converts the settings to a content page.
ToJson()	Converts the settings to JSON format.
TryGetSetting<T>(String, T)	Attempts to retrieve the value of a setting by its name. If the setting does not exist, it returns the default value provided.
Update(String)	Updates the settings of the command palette extension.

Settings.Add<T>(Setting<T>) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Adds a new setting to the command palette extension's configuration. This method is used to define settings that can be accessed and modified by the command palette extension.

Parameters

s [Setting<T>](#)

The setting to add. This should be an instance of the **Setting<T>** class, which contains the key, value, and other metadata for the setting.

Settings.GetSetting<T>(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **GetSetting** method retrieves the value of a specific setting from the command palette extension. This method is used to access the settings defined in the command palette extension's configuration.

Parameters

key String

The key of the setting to retrieve. This should match the key used when the setting was defined in the command palette extension.

Returns

An object of type **T** representing the value of the setting. The type **T** should match the type of the setting defined in the command palette extension's configuration. If the setting does not exist or cannot be converted to the specified type, an exception may be thrown.

Settings.ToContent() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToContent** method converts the settings of the command palette extension into a format suitable for display in the command palette. This is used to render the settings in a user-friendly manner.

Returns

An [IContent\[\]](#) array representing the settings in a format suitable for display in the command palette.

Settings.ToJson() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToJson** method converts the settings of the command palette extension into a JSON string representation. This is useful for serialization or for exporting the settings to a file or other storage format.

Returns

A **String** representing the settings in JSON format.

Settings.TryGetSetting<T>(String, T) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **TryGetSetting** method attempts to retrieve a setting value from the command palette extension's settings. If the setting is not found, it returns the default value provided.

Parameters

key String

The key of the setting to retrieve. This should be the name of the setting as defined in the extension's settings schema.

val T

The default value to return if the setting is not found. This value is used as a fallback in case the specified setting does not exist in the settings store.

Returns

A **Boolean** indicating whether the setting was found and successfully retrieved.

Settings.Update(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the settings of the command palette extension with the provided JSON string. This method is useful for applying changes to the settings without needing to manually set each property.

Parameters

data String

The JSON string containing the settings data to be updated.

SettingsForm Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [FormContent](#)

The **SettingsForm** class is a form that provides a user interface for configuring settings in the Command Palette extension. It allows users to input and modify settings through various controls such as text boxes, checkboxes, and dropdowns.

Methods

 Expand table

Method	Description
SubmitForm(String, String)	Submits the form data to the specified URL with the specified method.

SettingsForm.SubmitForm(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **SubmitForm** method is used to submit the form data to a specified URL using a specified HTTP method. This method is typically used to send user input from the form to a server or API for processing.

Parameters

inputs **String**

The inputs to be submitted with the form. This can include user-entered data, selections, or any other relevant information that needs to be sent to the destination.

data **String**

The data to be submitted with the form. This can include additional information or parameters that are required for the submission process.

Returns

An [ICommandResult](#) object that represents the result of the form submission. This object can contain information about the success or failure of the submission, as well as any relevant data returned from the server or API.

ShellHelpers Class


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)


The **ShellHelpers** class provides methods for opening commands in a shell environment. It allows you to execute commands with different user contexts, such as running as an administrator or as a different user.

Methods

 Expand table

Method	Description
OpenCommandInShell(String, String, String, String, ShellRunAsType, Boolean)	Opens a command in a shell environment with the specified parameters.
OpenInShell(String, String, String, ShellRunAsType, Boolean)	Opens a shell with the specified command and parameters.

Enums

 Expand table

Enum	Description
ShellRunAsType	The ShellRunAsType enumeration defines the different types of user contexts in which a command can be executed in the Command Palette. This enumeration is used to specify how a command should be run.

ShellHelpers.OpenCommandInShell(String, String, String, String, ShellRunAsType, Boolean) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **OpenCommandInShell** method opens a command in the shell with the specified parameters.

Parameters

path **String**

The path to the shell executable. This can be a full path or a relative path.

pattern **String**

The pattern to match the command. This can be a wildcard pattern or a regular expression.

arguments **String**

The arguments to pass to the shell. This can include options, flags, and other parameters.

workingDir **String**

The working directory for the shell. This is the directory where the command will be executed.

runAs [ShellRunAsType](#)

The user context in which to run the shell.

runWithHiddenWindow **Boolean**

The flag indicating whether to run the command in a hidden window.

Returns

A **Boolean** value indicating whether the command was successfully opened in the shell.

ShellHelpers.OpenInShell(String, String, String, **ShellRunAsType**, Boolean) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **OpenInShell** method opens a shell with the specified command and parameters. This method allows you to execute commands in a shell environment with different user contexts, such as running as an administrator or as a different user.

Parameters

path **String**

The path to the shell executable. This can be a full path or a relative path.

arguments **String**

The arguments to pass to the shell executable. This can include command-line options and parameters.

workingDir **String**

The working directory for the shell. This is the directory in which the shell will start executing commands.

runAs [ShellRunAsType](#)

The user context in which to run the shell.

runWithHiddenWindow **Boolean**

The flag indicating whether to run the shell with a hidden window. If set to `true`, the shell will not be visible to the user. This is useful for background tasks or when you don't want to interrupt the user's workflow.

Returns

A **Boolean** value indicating whether the shell was opened successfully.

ShellRunAsType Enum


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ShellRunAsType** enumeration defines the different types of user contexts in which a command can be executed in the Command Palette. This enumeration is used to specify how a command should be run.

Fields

 Expand table

Name	Description
Administrator	Indicates that the command should be run with elevated privileges (as an administrator).
None	Indicates that the command should be run with the current user's privileges (no elevation).
OtherUser	Indicates that the command should be run as a different user. This requires additional user input to specify the user context.

StatusMessage Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [IStatusMessage](#)

The **StatusMessage** class is used to represent a status message in the command palette. It provides properties to set the message text, progress state, and message state.

Properties

 Expand table

Property	Type	Description
Message	String	The text of the message. This property is used to display a message to the user in the command palette.
Progress	IProgressState	The progress state of the message. This property is used to indicate the progress of an operation.
State	MessageState	The state of the message. This property is used to indicate the status of the message, such as success or error.

StringMatcher Class


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)


The **StringMatcher** class provides methods for performing fuzzy matching and searching on strings. It is used to compare two strings and determine their similarity based on a specified precision score.

Constructors

 Expand table


Constructor	Description
StringMatcher()	Initializes a new instance of the StringMatcher class.

Properties

 Expand table

Property	Type	Description
Instance	StringMatcher	Gets the singleton instance of the StringMatcher class.
UserSettingSearchPrecision	SearchPrecisionScore	Gets or sets the precision score used for fuzzy matching.

Methods

 Expand table

Method	Description
FuzzyMatch(String, String)	Performs a fuzzy match between two strings.
FuzzyMatch(String, String,	Performs a fuzzy match between two strings with a

Method	Description
MatchOption)	specified match option.
FuzzySearch(String, String)	Performs a fuzzy search between two strings.

StringMatcher Constructors

Article • 03/31/2025

StringMatcher() Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [StringMatcher](#) class.

C#

```
public StringMatcher(/*IAlphabet alphabet = null*/)
{
    // _alphabet = alphabet;
}
```

StringMatcher.FuzzyMatch(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FuzzyMatch** method performs a fuzzy match between two strings to determine their similarity. It compares the *query* parameter with the *stringToCompare* parameter and returns a [MatchResult](#) object that indicates the result of the comparison.

Parameters

query String

The query to compare against.

stringToCompare String

The string to compare with the query.

Returns

A [MatchResult](#) object that indicates the result of the fuzzy match.

StringMatcher.FuzzyMatch(String, String, [MatchOption](#)) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FuzzyMatch** method performs a fuzzy match between two strings to determine their similarity. It compares the *query* parameter with the *stringToCompare* parameter and returns a [MatchResult](#) object that indicates the result of the comparison.

Parameters

query String

The string to be compared against the *stringToCompare* parameter.

stringToCompare String

The string to be compared with the *query* parameter.

opt [MatchOption](#)

The options that control the fuzzy matching behavior. This parameter allows you to specify how the fuzzy match should be performed, such as case sensitivity or other matching criteria.

Returns

A [MatchResult](#) object that contains the result of the fuzzy match.

StringMatcher.FuzzySearch(String, String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **FuzzySearch** method performs a fuzzy search between two strings to determine their similarity. It compares the *query* parameter with the *stringToCompare* parameter and returns a [MatchResult](#) object that indicates the result of the comparison.

Parameters

query String

The string to be compared against the *stringToCompare* string.

stringToCompare String

The string to be compared with the *query* string.

Returns

A [MatchResult](#) object that contains the result of the fuzzy search.

Tag Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [ITag](#)

The **Tag** class represents a tag that can be used in the command palette. It provides properties for customizing the appearance of the tag, including background color, foreground color, icon, text, and tooltip.

Constructors

 Expand table

Constructor	Description
Tag()	Initializes a new instance of the Tag class with default values.
Tag(String)	Initializes a new instance of the Tag class with a specified text.

Properties

 Expand table

Property	Type	Description
Background	OptionalColor	The background color of the tag.
Foreground	OptionalColor	The foreground color of the tag.
Icon	IIconInfo	The icon associated with the tag.
Text	String	The text displayed on the tag.
ToolTip	String	The tooltip text displayed when hovering over the tag.

Tag Constructors

Article • 03/31/2025

Tag() Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [Tag](#) class.

C#

```
public Tag()  
{  
}
```

Tag(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [Tag](#) class with its **_text** set to *text*.

C#

```
public Tag(string text)  
{  
    _text = text;  
}
```

Parameters

text String

The text to be set for the tag. This parameter allows you to initialize the tag with a specific string value at the time of its creation.

TextSetting Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Setting<String>](#)

The **TextSetting** class represents a setting that allows users to input text. It is a specialized type of setting that can be used in the Command Palette to capture user input in a text format. This class provides properties and methods to manage the text input, including support for multiline input and placeholder text.

Constructors

 Expand table


Constructor	Description
TextSetting(String, String)	This constructor initializes a new instance of the TextSetting class with a specified name and description.
TextSetting(String, String, String, String)	This constructor initializes a new instance of the TextSetting class with a specified name, description, default value, and placeholder text.

Properties

 Expand table

Property	Type	Description
Multiline	Boolean	Indicates whether the text input should support multiple lines.
Placeholder	String	The placeholder text that is displayed when the input field is empty.

Methods

 Expand table

Method	Description
LoadFromJson(JsonObject)	Loads the setting from a JSON object. This method is used to deserialize the setting from a JSON representation.
ToDictionary()	Converts the setting to a dictionary representation. This method is useful for serializing the setting for storage or transmission.
ToState()	Converts the setting to a state representation. This method is useful for capturing the current state of the setting for storage or transmission.
Update(JsonObject)	Updates the setting with values from a JSON object. This method is used to apply changes to the setting based on a JSON representation.

TextSetting Constructors

Article • 03/31/2025

The [TextSetting](#) class has constructors that allow you to create instances of the class with different parameters.

TextSetting(String, String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [TextSetting](#) class with a *key* and a *defaultValue*.

C#

```
public TextSetting(string key, string defaultValue)
    : base(key, defaultValue)
{
}
```

Parameters

key String

The unique identifier for the setting.

defaultValue String

The default value for the setting.

TextSetting(String, String, String, String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [TextSetting](#) class with a *key*, *label*, *description*, and *defaultValue*.

C#

```
public TextSetting(string key, string label, string description, string  
defaultValue)  
    : base(key, label, description, defaultValue)  
{  
}
```

Parameters

key **String**

The unique identifier for the setting.

label **String**

The display name of the setting.

description **String**

The description of the setting.

defaultValue **String**

The default value for the setting.

TextSetting.LoadFromJson(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LoadFromJson** method loads the settings from a *jsonObject* parameter into the current [TextSetting](#) instance.

Parameters

jsonObject **JsonObject**

The **JsonObject** that contains the settings to be loaded.

Returns

A [TextSetting](#) instance with the settings loaded from the provided **JsonObject**.

TextSetting.ToDictionary() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDictionary** method converts the current [TextSetting](#) instance into a dictionary representation.

Returns

A **Dictionary<String, Object>** that contains the settings in the current **TextSetting** instance.

TextSetting.ToState() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToState** method is used to convert the current state of a [TextSetting](#) instance into a string representation.

Returns

A **String** representing the current state of the setting. The state is typically a JSON string that captures the current values of the setting's properties.

TextSetting.Update(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the setting with values from a *payload* parameter. This method is used to apply changes to the setting based on a JSON representation.

Parameters

payload **JsonObject**

The **JsonObject** containing the values to update the setting. The **JsonObject** should contain the properties that correspond to the setting's fields.

ThumbnailHelper Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ThumbnailHelper** class provides methods to retrieve thumbnail images for commands in the Command Palette. It is used to enhance the visual representation of commands by providing associated images.

Methods

 Expand table

Method	Description
GetThumbnail(String)	Retrieves the thumbnail image for a command.

ThumbnailHelper.GetThumbnail(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ThumbnailHelper.GetThumbnail(String)** method retrieves the thumbnail image for a command in the Command Palette. This method is useful for enhancing the visual representation of commands by providing associated images.

Parameters

path String

The path to the command for which the thumbnail is requested.

Returns

A **Task<IRandomAccessStream>** that represents the thumbnail image for the specified command. If no thumbnail is found, the task will return `null`.

ToastArgs Class

Article • 03/31/2025


Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Implements [IToastArgs](#)

The **ToastArgs** class implements the **IToastArgs** interface and is used to define the arguments for a toast notification. It contains properties for the message text and the result of the command.

Properties

 Expand table

Property	Type	Description
Message	String	Gets or sets the text of the toast.
Result	ICommandResult	Gets or sets the result of the command. Defaults to CommandResult.Dismiss() .

ToastStatusMessage Class


Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToastStatusMessage** class is used to create and display toast notifications in Windows. It provides a way to show status messages to users in a non-intrusive manner.

Constructors

 Expand table


Constructor	Description
ToastStatusMessage(StatusMessage)	Creates a new instance of the ToastStatusMessage class with the specified status message.
ToastStatusMessage(String)	Creates a new instance of the ToastStatusMessage class with the specified message.

Properties

 Expand table

Property	Type	Description
Duration	Integer	The duration for which the toast message will be displayed.
Message	StatusMessage	The status message to be displayed in the toast notification.

Methods

 Expand table

Method	Description
Show()	Displays the toast notification with the specified message.

ToastStatusMessage Constructors

Article • 03/31/2025

ToastStatusMessage([StatusMessage](#)) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ToastStatusMessage](#) class with its [Message](#) property set to *message*.

C#

```
public ToastStatusMessage(StatusMessage message)
{
    Message = message;
}
```

Parameters

message [StatusMessage](#)

The **StatusMessage** to be displayed in the toast notification.

ToastStatusMessage(String) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ToastStatusMessage](#) class with its [Message](#) property set to *text*.

C#

```
public ToastStatusMessage(string text)
{
```

```
Message = new StatusMessage() { Message = text };  
}
```

Parameters

text **String**

The message to be displayed in the toast notification.

ToastStatusMessage.Show() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Show()** method is used to display the toast notification with the specified message.

ToggleSetting Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [Setting<Boolean>](#)


The **ToggleSetting** class represents a toggle setting in the Command Palette Extensions Toolkit. It is used to define settings that can be toggled on or off, such as enabling or disabling a feature.

Constructors

 [Expand table](#)

Constructor	Description
ToggleSetting(String, Boolean)	Creates a new instance of the ToggleSetting class with the specified name and default value.
ToggleSetting(String, String, String, Boolean)	Creates a new instance of the ToggleSetting class with the specified name, description, category, and default value.

Methods

 [Expand table](#)

Method	Description
LoadFromJson(JsonObject)	Loads the toggle setting from a JSON object.
ToDictionary()	Converts the toggle setting to a dictionary representation.
ToState()	Converts the toggle setting to a state representation.
Update(JsonObject)	Updates the toggle setting with the specified JSON object.

ToggleSetting Constructors

Article • 03/31/2025

ToggleSetting(String, Boolean) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ToggleSetting](#) class with a *key* and a *defaultValue*.

C#

```
public ToggleSetting(string key, bool defaultValue)
    : base(key, defaultValue)
{
}
```

Parameters

key String

The unique identifier for the toggle setting. This key is used to reference the setting in the application.

defaultValue Boolean

The default value of the toggle setting. This value is used when the setting is first created or when it is reset to its default state.

ToggleSetting(String, String, String, Boolean) Constructor

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Initializes a new instance of the [ToggleSetting](#) class with a *key*, *label*, *description*, and *defaultValue*.

C#

```
public ToggleSetting(string key, string label, string description, bool
defaultValue)
    : base(key, label, description, defaultValue)
{
}
```

Parameters

key **String**

The unique identifier for the toggle setting. This key is used to reference the setting in the application.

label **String**

The display name of the toggle setting. This label is shown to users in the user interface.

description **String**

The description of the toggle setting. This description provides additional information about the setting and its purpose.

defaultValue **Boolean**

The default value of the toggle setting. This value is used when the setting is first created or when it is reset to its default state.

ToggleSetting.LoadFromJson(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **LoadFromJson** method loads a toggle setting from a *JsonObject* parameter. This is used to initializing the toggle setting with data that has been serialized in JSON format.

Parameters

JsonObject **JsonObject**

The **JsonObject** that contains the toggle setting data. This object is typically loaded from a JSON file or string.

Returns

A [ToggleSetting](#) object that represents the loaded toggle setting.

ToggleSetting.ToDictionary() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToDictionary** method converts a toggle setting in the command palette extension to a dictionary representation.

Returns

A **Dictionary<String, Object>** that represents the current state of the toggle setting.

ToggleSetting.ToState() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **ToState** method retrieves the current state of a toggle setting in the command palette extension.

Returns

A **String** that represents the current state of the toggle setting.

ToggleSetting.Update(JsonObject) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Update** method updates the value of a toggle setting in the command palette extension.

Parameters

payload **JsonObject**

A **JsonObject** containing the new value for the setting.

TreeContent Class

Article • 03/31/2025

Definition


Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Inherits [BaseObservable](#)

Implements [ITreeContent](#)

The **TreeContent** class provides a way to manage tree-like structures in the Command Palette Extensions Toolkit. It allows for the organization of content in a hierarchical manner, making it easier to navigate and manage complex data structures.

Properties

 Expand table

Property	Type	Description
Children	IContent[]	The child content items of the tree node.
RootContent	IContent	The root content item of the tree.

Events

 Expand table

Event	Description
<code>Windows.Foundation.TypedEventHandler<object, IItemsChangedEventArgs> ItemsChanged</code>	Occurs when the items in the tree content change. This event is used to notify subscribers about changes in the tree structure, such as additions or removals of child items.

Methods

 Expand table

Method	Description
GetChildren()	Retrieves the child content items of the tree node. This method is used to access the children of a specific node in the tree structure.

TreeContent.Children() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Children** method retrieves the child items of the current tree content in the Command Palette. This method is used to populate the Command Palette with the items that are displayed when the user expands the tree content.

Returns

An [IContent\[\]](#) array representing the child items of the current tree content. This array contains the items that are displayed in the Command Palette when the user expands the tree content.

TreeContent.RaiseItemsChanged(Integer) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **RaiseItemsChanged** method is used to notify the Command Palette that the items in the tree content have changed. This method is typically called when the number of items in the tree content has changed, and it allows the Command Palette to update its display accordingly.

Parameters

totalItems **Integer**

An integer representing the total number of items in the tree content. This value is used to update the Command Palette with the new item count.

Utilities Class

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

The **Utilities** class provides utility methods for the Command Palette Extensions Toolkit.

Methods

 [Expand table](#)

Method	Description
BaseSettingsPath(String)	Used to produce a path to a settings folder which your app can use.
IsPackaged()	Can be used to quickly determine if this process is running with package identity.

Utilities.BaseSettingsPath(String) Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Used to produce a path to a settings folder which your app can use. This is useful for storing settings or other data that your app needs to persist between runs. The path returned will be different depending on whether your app is running packaged or not.

Parameters

settingsFolderName **String**

A **String** representing the name of the settings folder. This is the name of the folder that will be created in the settings path. The folder will be created if it does not already exist.

Returns

A **String** representing the path to the settings folder. If your app is running packaged, this will return the redirected local app data path (`Packages/{your_pfn}/LocalState`). If not, it'll return `%LOCALAPPDATA%\{settingsFolderName}`.

Utilities.IsPackaged() Method

Article • 03/31/2025

Definition

Namespace: [Microsoft.CommandPalette.Extensions.Toolkit](#)

Can be used to quickly determine if this process is running with package identity (returns `true`).

Returns

A **Boolean** value indicating whether the current process is running with package identity. It returns `true` if the process is running with package identity; otherwise, it returns `false`.