Windows Settings Plugin

The Windows settings Plugin allows users to search the Windows settings.

Special functions (differ from the regular functions)

- Support modern Windows settings (Windows 10+)
- Support legacy Windows settings (Windows 7, 8.1)
- Support extra programs for setting (like ODBC)
- Support search by the area of the setting (like `Privacy`)
- Support search for alternative names of a setting

How to add a new Windows Setting or change one

All Windows settings are located in `WindowsSettings.json` in root folder of the project. The `WindowsSettings.json` use a JSON schema file that make it easier to edit it.

Key	Optional	Value type	String prefix
`Name`	**No**	String	
`Type`	**No**	String	`App`
`Command`	**No**	String	
`Areas`	Yes	List with strings	`Area`
`AltNames`	Yes	List with strings	
`Note`	Yes	String	`Note`
`IntroducedInBuild`	Yes	Integer	
`DeprecatedInBuild`	Yes	Integer	
`ShowAsFirstResult`	Yes	Boolean	

A minimum entry for the `WindowsSettings.json` looks like:

```
{
  "Name": "mySetting",
  "Type": "AppSettingsApp",
  "Command": "ms-settings:mySetting"
}
```

A full entry for the `WindowsSettings.json` looks like:

```
{
  "Name": "mySetting",
  "Type": "AppSettingsApp",
  "Command": "ms-settings:mySetting",
  "Areas": [ "AreaMySettingArea" ],
  "AltNames": [ "NiceSetting" ],
  "Note": "NoteMySettingNote",
  "IntroducedInBuild" : 1903,
  "DeprecatedInBuild" : 2004,
  "ShowAsFirstResult" : true
}
```

Remarks

- The `Command` for modern Windows settings should start with `ms-settings:`
- The `Command` for legacy Windows settings should start with `control`
- The integer value for `IntroducedInBuild` and `DeprecatedInBuild` must be in range of `0` to `4294967295`
- The strings for `Name`, `AltNames`, `Areas`, `Type` and `Note` must not contain whitespace(s) or special characters (#, €, $, etc.)
- The strings for `Name`, `AltNames`, `Areas`, `Type` and `Note` are used as ids for the resource file under `Properties\Resources.resx`
- When you add new strings make sure you have add add all translations for it.

Scores

There are three different score types with different start values.

Score type	Start value
First result score	10500
High score	10000
Medium score	5000
Low score	1000

Each score will decreased by one when a condition match.

Priority	Condition	Score type
1.	Settings name starts with the search value	High score
2.	Settings name contain the search value	Medium score
3.	Setting has no area	Low score
4.	One area of the settings starts with the search value	Low score
5.	Setting has no alternative name	Low score
6.	One alternative name of the settings starts with the search value	Medium score
x.	no condition match	Low score

Remarks

- For each score condition we check if the property "ShowAsFirstResult" of the setting is true. If yes we use the firstResultScore instead of condition`s score.

Important for developers

General

- The assembly name is cached into `_assemblyName` (to avoid to many calls of `Assembly.GetExecutingAssembly()`)

Microsoft.PowerToys.Run.Plugin.WindowsSettings project

Important plugin values (meta-data)

Name	Value
ActionKeyword	`$`
ExecuteFileName	`Microsoft.PowerToys.Run.Plugin.WindowsSettings.dll`
ID	`5043CECEE6A748679CBE02D27D83747A`

Interfaces used by this plugin

The plugin use only these interfaces (all inside the `Main.cs`):

- `Wox.Plugin.IPlugin`
- `Wox.Plugin.IContextMenu`
- `Wox.Plugin.IPluginI18n`

Program files

File	Content
`Classes\WindowsSetting.cs`	A class that represent one Windows setting
`Classes\WindowsSettings.cs`	A wrapper class that only contains a list with Windows settings (see 1)
`Helper\ContextMenuHelper.cs`	All functions to build the context menu (for each result entry)
`Helper\JsonSettingsListHelper.cs`	All functions to load the windows settings from a JSON file
`Helper\ResultHelper.cs`	All functions to convert internal results into WOX results
`Helper\TranslationHelper.cs`	All functions to translate the result in the surface language
`Helper\UnsupportedSettingsHelper.cs`	All functions to filter not supported Windows settings out
`Helper\WindowsSettingsPathHelper.cs`	All functions to build the area paths
`Images\WindowsSettings.dark.png`	Symbol for the results for the dark theme
`Images\WindowsSettings.light.png`	Symbol for the results for the light theme

`Properties\Resources.Designer.resx`	File that contain all translatable keys
`Properties\Resources.resx`	File that contain all translatable strings in the neutral language
`GlobalSuppressions.cs`	Code suppressions (no real file, linked via *.csproj)
`Main.cs`	Main class, the only place that implement the WOX interfaces
`plugin.json`	All meta-data for this plugin
`StyleCop.json`	Code style (no real file, linked via *.csproj)

1. We need this extra wrapper class to make it possible that the JSON file can have and use a JSON schema file. Because the JSON file must have a object as root type, instead of a array.

Important project values (*.csproj)

Name	Value
TargetFramework	`net6.0-windows` (.NET 5) or `net6.0-windows10.0.18362.0` (OS version specific)
Platforms	`x64`
Output	`..\..\..\..\..\x64\Debug\modules\launcher\Plugins\Microsoft.PowerToys.Run.Plugin.Win`
RootNamespace	`Microsoft.PowerToys.Run.Plugin.WindowsSettings`
AssemblyName	`Microsoft.PowerToys.Run.Plugin.WindowsSettings`

Project dependencies

Packages

Package	Version
`StyleCop.Analyzers`	1.1.118

Projects

- `Wox.Infrastructure`
- `Wox.Plugin`