



We use optional cookies to improve your experience on our websites, such as through social media connections, and to display personalized advertising based on your online activity. If you reject optional cookies, only cookies necessary to provide you the services will be used. You may change your selection by clicking "Manage Cookies" at the bottom of the page. [Privacy Statement](#) [Third-Party Cookies](#)

Accept

Reject

Manage cookies



Learn

Discover ▾

Product documentation ▾

Development languages ▾

Topics ▾



Sign in



Application Registration

Article • 08/19/2021 • 7 contributors

[Feedback](#)

In this article

[Finding an Application Executable](#)

[Registering Applications](#)

[Registering Verbs and Other File Association Information](#)

[Registering a Perceived Type](#)

[Related topics](#)

This topic discusses how applications can expose information about themselves necessary to enable certain scenarios. This includes information needed to locate the application, the verbs that the application supports and the types of files that an application can handle.

This topic is organized as follows:

- [Finding an Application Executable](#)
- [Registering Applications](#)
 - [Using the App Paths Subkey](#)
 - [Using the Applications Subkey](#)
- [Registering Verbs and Other File Association Information](#)
- [Registering a Perceived Type](#)
- [Related topics](#)

ⓘ Note

Applications can also be registered in the Set Program Access and Defaults (SPAD) and Set Your Default Programs (SYDP) control panel applications. For information about SPAD and SYDP application registration, see [Guidelines for File Associations and Default Programs](#), and [Set Program Access and Defaults \(SPAD\)](#).

Finding an Application Executable

When the [ShellExecuteEx](#) function is called with the name of an executable file in its *lpFile* parameter, there are several places where the function looks for the file. We recommend registering your application in the **App Paths** registry subkey. Doing so avoids the need for applications to modify the system PATH environment variable.

The file is sought in the following locations:

- The current working directory.
- The **Windows** directory only (no subdirectories are searched).
- The **Windows\System32** directory.
- Directories listed in the PATH environment variable.
- Recommended:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Paths

Registering Applications

Both the **App Paths** and **Applications** registry subkeys are used to register and control the behavior of the system on behalf of applications. The **App Paths** subkey is the preferred location.

Using the App Paths Subkey

In Windows 7 and later, we strongly recommend you install applications per user rather than per machine. An application that is installed for per user can be registered under

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\App

Paths. An application that is installed for all users of the computer can be registered under **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\App Paths.**

The entries found under **App Paths** are used primarily for the following purposes:

- To map an application's executable file name to that file's fully qualified path.
- To pre-pend information to the PATH environment variable on a per-application, per-process basis.

If the name of a subkey of **App Paths** matches the file name, the Shell performs two actions:

- The (Default) entry is used as the file's fully qualified path.
- The Path entry for that subkey is pre-pended to the PATH environment variable of that process. If this is not required, the Path value can be omitted.

Potential issues to be aware of include:

- The Shell limits the length of a command line to $\text{MAX_PATH} * 2$ characters. If there are many files listed as registry entries or their paths are long, file names later in the list could be lost as the command line is truncated.
- Some applications do not accept multiple file names in a command line.
- Some applications that accept multiple file names do not recognize the format in which the Shell provides them. The Shell provides the parameter list as a quoted string, but some applications might require strings without quotes.
- Not all items that can be dragged are part of the file system; for example, printers. These items do not have a standard Win32 path, so there is no way to provide a meaningful *lpParameters* value to [ShellExecuteEx](#).

Using the DropTarget entry avoids these potential issues by providing access to all of the clipboard formats, including [CFSTR_SHELLIDLIST](#) (for long file lists) and [CFSTR_FILECONTENTS](#) (for non-file-system objects).

To register and control the behavior of your applications with the App Paths subkey:

1. Add a subkey with the same name as your executable file to the **App Paths** subkey, as shown in the following registry entry.

```
HKEY_LOCAL_MACHINE or HKEY_CURRENT_USER
SOFTWARE
  Microsoft
    Windows
      CurrentVersion
        App Paths
          file.exe
            (Default)
            DontUseDesktopChangeRouter
            DropTarget
            Path
            UseUrl
```

2. See the following table for details of the **App Paths** subkey entries.

 Expand table

| Registry entry | Details |
|----------------------------|---|
| (Default) | Is the fully qualified path to the application. The application name provided in the (Default) entry can be stated with or without its .exe extension. If necessary, the ShellExecuteEx function adds the extension when searching App Paths subkey. The entry is of the REG_SZ type. |
| DontUseDesktopChangeRouter | Is mandatory for debugger applications to avoid file dialog deadlocks when debugging the Windows Explorer process. Setting the DontUseDesktopChangeRouter entry produces a slightly less efficient handling of the change notifications, however. The entry is of the REG_DWORD type and the value is 0x1. |
| DropTarget | Is a class identifier (CLSID). The DropTarget entry contains the CLSID of an object (usually a local server rather than an in-process server) that implements IDropTarget . By default, when the drop target is an executable file, and no DropTarget value is provided, the Shell converts the list of dropped files into a command-line parameter and passes it to ShellExecuteEx through <i>lpParameters</i> . |

| | |
|--------------------|--|
| Path | Supplies a string (in the form of a semicolon-separated list of directories) to append to the PATH environment variable when an application is launched by calling ShellExecuteEx . It is the fully qualified path to the .exe. It is of REG_SZ . In Windows 7 and later , the type can be REG_EXPAND_SZ , and is commonly REG_EXPAND_SZ %ProgramFiles% . Note: In addition to the (Default), Path, and DropTarget entries recognized by the Shell, an application can also add custom values to its executable file's App Paths subkey. We encourage application developers to use the App Paths subkey to provide an application-specific path instead of making additions to the global system path. |
| SupportedProtocols | Creates a string that contains the URL protocol schemes for a given key. This can contain multiple registry values to indicate which schemes are supported. This string follows the format of scheme1:scheme2 . If this list is not empty, file: will be added to the string. This protocol is implicitly supported when <i>SupportedProtocols</i> is defined. |

| | |
|--------|--|
| UseUrl | <p>Indicates that your application can accept a URL (instead of a file name) on the command line. Applications that can open documents directly from the internet, like web browsers and media players, should set this entry.</p> <p>When the ShellExecuteEx function starts an application and the UseUrl=1 value is not set, ShellExecuteEx downloads the document to a local file and invokes the handler on the local copy. For example, if the application has this entry set and a user right-clicks on a file stored on a web server, the Open verb will be made available. If not, the user will have to download the file and open the local copy.</p> <p>The UseUrl entry is of REG_DWORD type, and the value is 0x1.</p> <p>In Windows Vista and earlier, this entry indicated that the URL should be passed to the application along with a local file name, when called via ShellExecuteEx. In Windows 7, it indicates that the application can understand any http or https url that is passed to it, without having to supply the cache file name as well. This registry key is associated with the <i>SupportedProtocols</i> key.</p> |
|--------|--|

Using the Applications Subkey

Through the inclusion of registry entries under the **HKEY_CLASSES_ROOT\Applications\ApplicationName.exe** subkey, applications can provide the application-specific information shown in the following table.

 Expand table

| Registry entry | Description |
|----------------|---|
| shell\verb | <p>Provides the verb method for calling the application from OpenWith. Without a verb definition specified here, the system assumes that the application supports CreateProcess, and passes the file name on the command line. This functionality applies to all the verb methods, including DropTarget, ExecuteCommand, and Dynamic Data Exchange (DDE).</p> |

| | |
|-----------------|--|
| DefaultIcon | Enables an application to provide a specific icon to represent the application instead of the first icon stored in the .exe file. |
| FriendlyAppName | Provides a way to get a localizable name to display for an application instead of just the version information appearing, which may not be localizable. The association query ASSOCSTR reads this registry entry value and falls back to use the FileDescription name in the version information. If that name is missing, the association query defaults to the display name of the file. Applications should use ASSOCSTR_FRIENDLYAPPNAME to retrieve this information to obtain the proper behavior. |
| SupportedTypes | Lists the file types that the application supports. Doing so enables the application to be listed in the cascade menu of the Open with dialog box. |
| NoOpenWith | Indicates that no application is specified for opening this file type. Be aware that if an OpenWithProgIDs subkey has been set for an application by file type, and the ProgID subkey itself does not also have a NoOpenWith entry, that application will appear in the list of recommended or available applications even if it has specified the NoOpenWith entry. For more information, see How to Include an Application in the Open With Dialog Box and How to exclude an Application from the Open with Dialog Box . |
| IsHostApp | Indicates that the process is a host process, such as Rundll32.exe or Dllhost.exe, and should not be considered for Start menu pinning or inclusion in the Most Frequently Used (MFU) list. When launched with a shortcut that contains a non-null argument list or an explicit Application User Model IDs (AppUserModelIDs) , the process can be pinned (as that shortcut). Such shortcuts are candidates for inclusion in the MFU list. |
| NoStartPage | Indicates that the application executable and shortcuts should be excluded from the Start menu and from pinning or inclusion in the MFU list. This entry is typically used to exclude system tools, installers and uninstallers, and readme files. |

| | |
|----------------------------------|---|
| UseExecutableForTaskbarGroupIcon | Causes the taskbar to use the default icon of this executable if there is no pinnable shortcut for this application, and instead of the icon of the window that was first encountered. |
| TaskbarGroupIcon | Specifies the icon used to override the taskbar icon. The window icon is normally used for the taskbar. Setting the TaskbarGroupIcon entry causes the system to use the icon from the .exe for the application instead. |

Examples

Some examples of application registrations through the **HKEY_CLASSES_ROOT\Applications\ApplicationName.exe** subkey are as follows. All registry entry values are of **REG_SZ** type, with the exception of **DefaultIcon** which is of **REG_EXPAND_SZ** type.

```
HKEY_CLASSES_ROOT
  Applications
    wordpad.exe
      FriendlyAppName = @%SystemRoot%\System32\shell32.dll,-2206
```

```
HKEY_CLASSES_ROOT
  Applications
    wmpplayer.exe
      SupportedTypes
        .3gp2
```

```
HKEY_CLASSES_ROOT
  Applications
    wmpplayer.exe
      DefaultIcon
        (Default) = %SystemRoot%\system32\wmploc.dll,-730
```



```
HKEY_CLASSES_ROOT
  Applications
    WScript.exe
      NoOpenWith
```

```
HKEY_CLASSES_ROOT
  Applications
    photoviewer.dll
      shell
        open
          DropTarget
            Clsid = {FFE2A43C-56B9-4bf5-9A79-CC6D4285608A}
```

```
HKEY_CLASSES_ROOT
  Applications
    mspaint.exe
      SupportedTypes
        .bmp
        .dib
        .rle
        .jpg
        .jpeg
        .jpe
        .jfif
        .gif
        .emf
        .wmf
        .tif
        .tiff
        .png
        .ico
```

Registering Verbs and Other File Association Information

Subkeys registered under **HKEY_CLASSES_ROOT\SystemFileAssociations** enable the Shell to define the default behavior of attributes for file types and enable shared file associations. When users change the default application for a file type, the ProgID of the new default application has priority in providing verbs and other association information. This priority is due to it being the first

entry in the association array. If the default program is changed, the information under the previous ProgID is no longer available.

To deal proactively with the consequences of a change to default programs, you can use **HKEY_CLASSES_ROOT\SystemFileAssociations** to register verbs and other association information. Due to their location after the ProgID in the association array, these registrations are lower priority. These SystemFileAssociationsregistrations are stable even when users change the default programs, and provide a location to register secondary verbs that will always be available for a particular file type. For a registry example, see [Registering a Perceived Type](#) later in this topic.

The following registry example shows what happens when the user runs the **Default Programs** item in Control Panel to change the default for .mp3 files to App2ProgID. After changing the default, Verb1 is no longer available, and Verb2 becomes the default.

```
HKEY_CLASSES_ROOT
    .mp3
        (Default) = App1ProgID
```

```
HKEY_CLASSES_ROOT
    App1ProgID
        shell
            Verb1
```

```
HKEY_CLASSES_ROOT
    App2ProgID
        shell
            Verb2
```

Registering a Perceived Type

Registry values for perceived types are defined as subkeys of the **HKEY_CLASSES_ROOT\SystemFileAssociations** registry subkey. For example,

the perceived type **text** is registered as follows:

```
HKEY_CLASSES_ROOT
  SystemFileAssociations
    text
      shell
        edit
          command
            (Default) = "%SystemRoot%\system32\notepad.exe"
        open
          command
            (Default) = "%SystemRoot%\system32\notepad.exe"
```

A file type's perceived type is indicated by including a `PerceivedType` value in the file type's subkey. The `PerceivedType` value is set to the name of the perceived type registered under **HKEY_CLASSES_ROOT\SystemFileAssociations** registry subkey, as shown in the previous registry example. To declare .cpp files as being of perceived type "text", for example, add the following registry entry:

```
HKEY_CLASSES_ROOT
  .cpp
    PerceivedType = text
```

Related topics

[File Types](#)

[How File Associations Work](#)

[Content View By File Type or Kind](#)

[File Type Verifier](#)

[File Type Handlers](#)

[Programmatic Identifiers](#)

[Perceived Types](#)

Association Arrays


Feedback

Was this page helpful?



 Yes

 No

[Provide product feedback](#)  | [Get help at Microsoft Q&A](#)

 English (United States)

  Your Privacy Choices


 Theme 

[Manage cookies](#)


[Previous Versions](#)

[Blog](#) 

[Contribute](#)

[Privacy](#) 

[Terms of Use](#)

[Trademarks](#) 

© Microsoft 2024