python

PowerShell

FPDF Library

PDF generator

Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Invoke-CommandInDesktopPackage'

PS C:\Users\wahid> Get-Help Invoke-CommandInDesktopPackage

NAME

Invoke-CommandInDesktopPackage

SYNOPSIS

A debugging tool that creates a new process in the context of a packaged app.

SYNTAX

Invoke-CommandInDesktopPackage [-PackageFamilyName] <System.String> [-AppId] <System.String> [-Command] <System.String> [[-Args] <System.String>] [[-PreventBreakaway]] [<CommonParameters>]

DESCRIPTION

`Invoke-CommandInDesktopPackage` creates a new process in the context of the supplied PackageFamilyName and AppId .

The created process will have the identity of the provided AppId and will have access to its virtualized file system and registry (if any). The new process will have a token that's similar to, but not identical to, a real AppId process.

The primary use-case of this command is to invoke debugging or troubleshooting tools in the context of the packaged app to access its virtualized resources. For example, you can run the Registry Editor to see virtualized registry keys, or Notepad to read virtualized files. See the important note that follows on using tools such as the Registry Editor that require elevation.

No guarantees are made about the behavior of the created process, other than it having the package identity and access to the package's virtualized resources. In particular, the new process will not be created in an AppContainer even if an AppId process would normally be created in an AppContainer. Features such as Privacy Controls or other App Settings may or may not apply to the new process. You shouldn't rely on any specific side-effects of using this command, as they're undefined and subject to change.

PARAMETERS

-AppId <System.String>

AppId is the Application ID from the target package's manifest.

For example, `MyAppName` is the Application ID in this manifest snippet:

`<Application Id="MyAppName" ... />`

-Args <System.String>

Optional arguments to be passed to the new process. For example, `/foo /bar`.

-Command <System.String>

An executable to invoke, like `regedit.exe`.

Note that if the executable requires elevation (like `regedit`), you must call `Invoke-CommandInDesktopPackage` from an already-elevated context.

Calling `Invoke-CommandInDesktopPackage` from a non-elevated context doesn't work as expected. The new process is created without the package context, and the PowerShell command fails.

-PackageFamilyName <System.String>
The Package Family Name of the target package. You can retrieve this by calling Get-AppxPackage (./Get-AppxPackage.md).

-PreventBreakaway <System.Management.Automation.SwitchParameter>
Causes all child processes of the invoked process to also be created in the context of the AppId . By default, child processes are created without any context. This switch is useful for running `cmd.exe` so that you can launch multiple other tools in the package context.

<CommonParameters>
This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

----- Example 1: Invoke Notepad to read virtualized files -----

```
$params = @{
    AppId             = 'ContosoApp'
    PackageFamilyName = 'Contoso.MyApp_abcdefgh23456'
    Command           = 'notepad.exe'
}
Invoke-CommandInDesktopPackage @params
```

REMARKS
To see the examples, type: "get-help Invoke-CommandInDesktopPackage -examples".
For more information, type: "get-help Invoke-CommandInDesktopPackage

-detailed".

For technical information, type: "get-help Invoke-CommandInDesktopPackage

-full".

For online help, type: "get-help Invoke-CommandInDesktopPackage -online"