How-to: Write an Advanced Function

An *Advanced* Function is one that contains either a `[cmdletbinding()]` attribute or the `Parameter` attribute, or both. An advanced function gains many of the standard cmdlet features such as common parameters (`-verbose, -whatif, -confirm`)

Below is a template for an Advanced Function, containing help text, parameters and begin-process-end blocks:

```
Function FnNameVERB-NOUN {
<#
 Add some comment based help for the function here.
.SYNOPSIS
    This advanced function does...
#>
    [CmdletBinding()]  # Add cmdlet features.
    Param (
        # Define parameters below, each separated by a comma

        [Parameter(Mandatory=$True)]
        [int]$DemoParam1,

        [Parameter(Mandatory=$False)]
        [ValidateSet('Alpha','Beta','Gamma')]
        [string]$DemoParam2,

        # you don't have to use the full [Parameter()] decorator on every p
        [string]$DemoParam3,

        [string]$DemoParam4, $DemoParam5

        # Add additional parameters here.
    )



    Begin {
        # Start of the BEGIN block.
        Write-Verbose -Message "Entering the BEGIN block [$($MyInvocation.M

        # Add additional code here.

    } # End Begin block

    Process {
        # Start of PROCESS block.
        Write-Verbose -Message "Entering the PROCESS block [$($MyInvocation


        # Add additional code here.


    } # End of PROCESS block.

    End {
        # Start of END block.
        Write-Verbose -Message "Entering the END block [$($MyInvocation.MyC
```

```
        # Add additional code here.

    } # End of the END Block.
} # End Function

#----------------[ Main ]-----------------
# Optional Script Execution goes here
# and can call any of the functions above.
```

Below is a more detailed list of all the options available for an Advanced Function.

Attributes

Several (optional) attributes can be added to the `Param` clause of an advanced function. The `Param` keyword is used to define parameters in both simple and advanced functions, The `Parameter` attribute in advanced functions is to validate (limit the valid values of each parameter).

`[parameter(Argument=value)]` # Define attributes for each parameter, see below for a list of Parameter Arguments.
`[alias("CN","MachineName")]` # Establish an alternate name(s) for the parameter.
`[AllowNull()]`
`[AllowEmptyString()]`
`[AllowEmptyCollection()]`
`[ValidateCount(1,5)]` # The number or parameters accepted.
`[ValidateLength(1,10)]` # minimum and max. number of characters in a parameter or variable value.
`[ValidatePattern("[0-9][0-9][0-9][0-9]")]` # specifies a regex that is compared to the parameter or variable value.
`[ValidateRange(0,10)]` # specifies a numeric range for each parameter or variable value.
`[ValidateScript({$_ -ge (get-date)})]` # specifies a script that is used to validate a parameter or variable value. `[ValidateSet("Low", "Average", "High")]`# specifies a set of valid values for a parameter or variable.
`[ValidateNotNull()]` # specifies that the parameter value cannot be null ($null).
`[ValidateNotNullOrEmpty()]` # specifies that the parameter value cannot be null and cannot be an empty string ("")
`DynamicParam {<statement-list>}` # A dynamic parameter.
`[Switch]<ParameterName>` # Set a parameter with only a True/False value.

Parameter arguments

These can be included in each `[Parameter( )]` attribute. The Parameter attribute is optional, and you can omit it if none of the parameters of your advanced function need attributes. Separate multiple arguments with commas.

`Mandatory=$true` # Whether the parameter is mandatory or optional (the default) If you call a function without passing a value for a mandatory parameter, PowerShell will prompt for the value.
`Position=0` # By default, PowerShell assigns each parameter a position number in the order declared, starting from 0.
`ParameterSetName="User"` # allow a function to accept different sets of parameters for different tasks.
`ValueFromPipeline=$true` # Accept values via the pipeline.
`ValueFromPipelineByPropertyName=$true` # Accept values via the pipeline of the

same type expected by the parameter and which also must have the same name as the parameter accepting pipeline input.
`ValueFromRemainingArguments=$true`
`HelpMessage="Custom prompt message that appears for a mandatory parameter."`

"If you want to go fast, go alone,
If you want to go far, go together" ~ African proverb

Related PowerShell Cmdlets:

Template - A simple template for new PowerShell scripts.
Microsoft docs: Parameters & About Functions Advanced Parameters
`$PSCmdlet` variable
The ParameterSetName property allows you to see the parameter set that is being used. Parameter sets allow you to create a function that performs different tasks based on the parameters that are specified when the function is run.
Scriptblock - A collection of statements.
Ref vars - Passing a reference variable to a function.