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

PowerShell Get-Help on command 'Write-Error'

PS C:\Users\wahid> Get-Help Write-Error

NAME

Write-Error

SYNOPSIS

Writes an object to the error stream.

SYNTAX

Write-Error [[-Message] <System.String>] [-Category {NotSpecified | OpenError | CloseError | DeviceError | DeadlockDetected | InvalidArgument | InvalidData | InvalidOperation | InvalidResult | InvalidType | MetadataError | NotImplemented | NotInstalled | ObjectNotFound | OperationStopped | OperationTimeout | SyntaxError | ParserError | PermissionDenied | ResourceBusy | ResourceExists | ResourceUnavailable | ReadError | WriteError | FromStdErr | SecurityError | ProtocolError | ConnectionError | AuthenticationError | LimitsExceeded | QuotaExceeded | NotEnabled}] [-CategoryActivity <System.String>] [-CategoryReason <System.String>] [-CategoryTargetName <System.String>] [-CategoryTargetType <System.String>] [-ErrorId <System.String>] [-RecommendedAction <System.String>] [-TargetObject <System.Object>] [<CommonParameters>]

Write-Error [[-Message] <System.String>] [-Category {NotSpecified | OpenError | CloseError | DeviceError | DeadlockDetected | InvalidArgument | InvalidData | InvalidOperation | InvalidResult | InvalidType | MetadataError | NotImplemented | NotInstalled | ObjectNotFound | OperationStopped | OperationTimeout | SyntaxError | ParserError | PermissionDenied | ResourceBusy | ResourceExists | ResourceUnavailable | ReadError | WriteError | FromStdErr | SecurityError | ProtocolError | ConnectionError | AuthenticationError | LimitsExceeded | QuotaExceeded | NotEnabled}] [-CategoryActivity <System.String>] [-CategoryReason <System.String>] [-CategoryTargetName <System.String>] [-CategoryTargetType <System.String>] [-ErrorId <System.String>] -Exception <System.Exception> [-RecommendedAction <System.String>] [-TargetObject <System.Object>] [<CommonParameters>]

Write-Error [-CategoryActivity <System.String>] [-CategoryReason <System.String>] [-CategoryTargetName <System.String>] [-CategoryTargetType <System.String>] -ErrorRecord <System.Management.Automation.ErrorRecord> [-RecommendedAction <System.String>] [<CommonParameters>]

DESCRIPTION

The `Write-Error` cmdlet declares a non-terminating error. By default, errors are sent in the error stream to the host program to be displayed, along with output.

To write a non-terminating error, enter an error message string, an ErrorRecord object, or an Exception object. Use the other parameters of `Write-Error` to populate the error record.

Non-terminating errors write an error to the error stream, but they don't stop command processing. If a non-terminating error is declared on one item in a collection of input items, the command continues to process the other items in the collection.

To declare a terminating error, use the `Throw` keyword. For more information,
see about_Throw (../Microsoft.PowerShell.Core/About/about_Throw.md).

PARAMETERS

-Category <System.Management.Automation.ErrorCategory>

Specifies the category of the error. The default value is NotSpecified .

The acceptable values for this parameter are:

- NotSpecified
- OpenError
- CloseError
- DeviceError
- DeadlockDetected
- InvalidArgument
- InvalidData
- InvalidOperation
- InvalidResult
- InvalidType
- MetadataError
- NotImplemented

- NotInstalled

- ObjectNotFound

- OperationStopped

- OperationTimeout

- SyntaxError

- ParserError

- PermissionDenied

- ResourceBusy

- ResourceExists

- ResourceUnavailable

- ReadError

- WriteError

- FromStdErr

- SecurityError

- ProtocolError

- ConnectionError

- AuthenticationError

- LimitsExceeded

- QuotaExceeded

- NotEnabled

For information about the error categories, see ErrorCategory Enumeration (https://go.microsoft.com/fwlink/?LinkId=143600).

-CategoryActivity <System.String>
Specifies the action that caused the error.

-CategoryReason <System.String>
Specifies how or why the activity caused the error.

-CategoryTargetName <System.String>
Specifies the name of the object that was being processed when the error occurred.

-CategoryTargetType <System.String>
Specifies the type of the object that was being processed when the error occurred.

-ErrorId <System.String>
Specifies an ID string to identify the error. The string should be unique to the error.

-ErrorRecord <System.Management.Automation.ErrorRecord>
Specifies an error record object that represents the error. Use the properties of the object to describe the error.

To create an error record object, use the `New-Object` cmdlet or get an error record object from the array in the `$Error` automatic variable.

-Exception <System.Exception>

Specifies an exception object that represents the error. Use the properties of the object to describe the error.

To create an exception object, use a hash table or use the `New-Object` cmdlet.

-Message <System.String>

Specifies the message text of the error. If the text includes spaces or special characters, enclose it in quotation marks. You can also pipe a message string to `Write-Error`.

-RecommendedAction <System.String>

Specifies the action that the user should take to resolve or prevent the error.

-TargetObject <System.Object>

Specifies the object that was being processed when the error occurred. Enter the object, a variable that contains the object, or a command that gets the object.

<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: Write an error for RegistryKey object -------

```
Get-ChildItem | ForEach-Object {
    if ($_.GetType().ToString() -eq "Microsoft.Win32.RegistryKey")
    {
        Write-Error "Invalid object" -ErrorId B1 -TargetObject $_
    }
    else
    {
        $_
    }
}
```

This command declares a non-terminating error when the `Get-ChildItem` cmdlet returns a `Microsoft.Win32.RegistryKey` object, such as the objects in the `HKLM:` or `HKCU:` drives of the PowerShell Registry provider.

------- Example 2: Write an error message to the console -------

```
Write-Error "Access denied."
```

This command declares a non-terminating error and writes an "Access denied" error. The command uses the Message parameter to specify the message, but omits the optional Message parameter name.

Example 3: Write an error to the console and specify the category

```
Write-Error -Message "Error: Too many input values." -Category InvalidArgument
```

This command declares a non-terminating error and specifies an error category.

----- Example 4: Write an error using an Exception object -----

```
$E = [System.Exception]@{Source="Get-ParameterNames.ps1";HelpLink="https://go.m
icrosoft.com/fwlink/?LinkID=113425"}
Write-Error -Exception $E -Message "Files not found. The $Files location
doesn't contain any XML files."
```

This command uses an Exception object to declare a non-terminating error.

The first command uses a hash table to create the System.Exception object. It saves the exception object in the `$E` variable. You can use a hash table to create any object of a type that has a null constructor.

The second command uses the `Write-Error` cmdlet to declare a non-terminating error. The value of the Exception parameter is the Exception object in the `$E` variable.

REMARKS

To see the examples, type: "get-help Write-Error -examples".

For more information, type: "get-help Write-Error -detailed".

For technical information, type: "get-help Write-Error -full".

For online help, type: "get-help Write-Error -online"