MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'Update-List'

PS:\>Get-HELP Update-List -Full

NAME

Update-List

SYNOPSIS

Adds items to and removes items from a property value that contains a collection of objects.

SYNTAX

Update-List [[-Property] <System.String>] [-Add <System.Object[]>] [-InputObject <System.Management.Automation.PSObject>] [-Remove <System.Object[]>]

[<CommonParameters>]

Update-List [[-Property] <System.String>] [-InputObject <System.Management.Automation.PSObject>] -Replace <System.Object[]> [<CommonParameters>]

DESCRIPTION

The `Update-List` cmdlet adds, removes, or replaces items in a property value of an object and returns the updated object. This cmdlet is designed for properties that

contain collections of objects.

The Add and Remove parameters add individual items to and remove them from the collection. The Replace parameter replaces the entire collection.

If you don't specify a property in the command, `Update-List` returns a hashtable that describes the update instead of updating the object. Later, you can use this
change set to update a list object.

This cmdlet works only when the property that's being updated supports the IList interface that `Update-List` uses. Also, any `Set` cmdlets that accept an update must
support the IList interface.

PARAMETERS

-Add <System.Object[]>

Specifies the property values to be added to the collection. Enter the values in the order that they should appear in the collection.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       False
Accept wildcard characters?  false
```

-InputObject <System.Management.Automation.PSObject>

Specifies the objects to be updated. You can also pipe the object to be updated to `Update-List`.

```
Required?                    false
Position?                    named
Default value                None
Accept pipeline input?       True (ByValue)
Accept wildcard characters?  false
```

```
    -Property <System.String>
        Specifies the property that contains the collection that's being updated. If you omit this parameter, `Update-List` returns
an object that represents the change
        instead of changing the object.

        Required?                    false
        Position?                    0
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -Remove <System.Object[]>
        Specifies the property values to be removed from the collection.

        Required?                    false
        Position?                    named
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false

    -Replace <System.Object[]>
        Specifies a new collection. This parameter replaces all items in the original collection with the items specified by this
parameter.

        Required?                    true
        Position?                    named
        Default value                None
        Accept pipeline input?       False
        Accept wildcard characters?  false

    <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).

INPUTS

    System.Management.Automation.PSObject

        You can pipe the object to be updated to this cmdlet.

OUTPUTS

    System.Collections.Hashtable

        By default, this cmdlet returns a hashtable that describes the update.

    System.Object

        When you specify the Property parameter, this cmdlet returns the updated object.

NOTES

    ----------- Example 1: Add items to a property value -----------

```
class Cards {

    [System.Collections.Generic.List[string]]$cards
    [string]$name

    Cards([string]$_name) {
        $this.name = $_name
        $this.cards = [System.Collections.Generic.List[string]]::new()
    }
```

```
        NewDeck() {
            $_suits = [char]0x2663,[char]0x2666,[char]0x2665,[char]0x2660
            $_values = 'A',2,3,4,5,6,7,8,9,10,'J','Q','K'
            $_deck = foreach ($s in $_suits){ foreach ($v in $_values){ "$v$s"} }
            $this | Update-List -Property cards -Add $_deck | Out-Null
        }

        Show() {
            Write-Host
            Write-Host $this.name ": " $this.cards[0..12]
            if ($this.cards.count -gt 13) {
                Write-Host (' ' * ($this.name.length+3)) $this.cards[13..25]
            }
            if ($this.cards.count -gt 26) {
                Write-Host (' ' * ($this.name.length+3)) $this.cards[26..38]
            }
            if ($this.cards.count -gt 39) {
                Write-Host (' ' * ($this.name.length+3)) $this.cards[39..51]
            }
        }

        Shuffle() { $this.cards = Get-Random -InputObject $this.cards -Count 52 }

        Sort() { $this.cards.Sort() }
    }
```

> [!NOTE] > The `Update-List` cmdlet outputs the updated object to the pipeline. We pipe the output to > `Out-Null` to suppress the unwanted display.

--- Example 2: Add and remove items of a collection property ---

```
    $player1 = [Cards]::new('Player 1')
    $player2 = [Cards]::new('Player 2')
```

```
$deck = [Cards]::new('Deck')
$deck.NewDeck()
$deck.Shuffle()
$deck.Show()

# Deal two hands
$player1 | Update-List -Property cards -Add $deck.cards[0,2,4,6,8] | Out-Null
$player2 | Update-List -Property cards -Add $deck.cards[1,3,5,7,9] | Out-Null
$deck | Update-List -Property cards -Remove $player1.cards | Out-Null
$deck | Update-List -Property cards -Remove $player2.cards | Out-Null

$player1.Show()
$player2.Show()
$deck.Show()

Deck :  4  7  J  5  A  8  J  Q  6  3  9  6  2
        K  4  10  8  10  9  6  K  7  3  Q  A  Q
        3  5  2  5  J  J  10  4  Q  10  4  2  2
        6  7  A  5  8  9  K  7  3  9  A  K  8

Player 1 :  4  J  A  J  6

Player 2 :  7  5  8  Q  3

Deck :  9  6  2  K  4  10  8  10  9  6  K  7  3
        Q  A  Q  3  5  2  5  J  J  10  4  Q  10
        4  2  2  6  7  A  5  8  9  K  7  3  9
        A  K  8
```

The output shows the state of the deck before the cards were dealt to the players. You can see that each player received five cards from the deck. The final output

shows the state of the deck after dealing the cards to the players. `Update-List` was used to select the cards from the deck and add them to the players' collection.

Then the players' cards were removed from the deck using `Update-List`.

----- Example 3: Add and remove items in a single command -----

```
# Player 1 wants two new cards - remove 2 cards & add 2 cards

$player1 | Update-List -Property cards -Remove $player1.cards[0,4] -Add $deck.cards[0..1] | Out-Null

$player1.Show()


# remove dealt cards from deck

$deck | Update-List -Property cards -Remove $deck.cards[0..1] | Out-Null

$deck.Show()


Player 1 :  J  A  J  9  6


Deck :  2  K  4  10  8  10  9  6  K  7  3  Q  A

        Q  3  5  2  5  J  J  10  4  Q  10  4  2

        2  6  7  A  5  8  9  K  7  3  9  A  K

        8
```

-------- Example 4: Apply a change set to a list object --------

```
$list = [System.Collections.ArrayList] (1, 43, 2)

$changeInstructions = Update-List -Remove 43 -Add 42

$changeInstructions


Name                           Value

----                           -----

Add                            {42}

Remove                         {43}


([PSListModifier]($changeInstructions)).ApplyTo($list)

$list
```

```
    1
    2
    42
```

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/update-list?view=powershell-5.1&WT.mc_id=ps-gethelp

Select-Object