PSProjectStatus Help[1]

https://jdhitsolutions.github.io

v0.16.0

Contents

PSProjectStatus

This PowerShell module is designed to make it easier to manage your projects and modules. It provides a snapshot overview of the project's status. You can use this to quickly determine when you last worked on a module and what high-level tasks remain. Status information is stored in a JSON file that resides in the module's root directory. If you have initialized *git* for the module, the project status will include the current branch.

Installation

Install this module from the PowerShell Gallery.

```
Install-Module PSProjectStatus
```

Or you can use the Microsoft.PowerShell.PSResourceGet module.

```
Install-PSResource PSProjectStatus -Scope AllUsers
```

This module is supported in Windows PowerShell 5.1 and PowerShell 7.

Module Commands

Status

- New-PSProjectStatus
- Get-PSProjectStatus
- Set-PSProjectStatus

Tasks

- New-PSProjectTask
- Get-PSProjectTask
- Remove-PSProjectTask

Other

- Get-PSProjectReport
- Get-PSProjectGitStatus
- Open-PSProjectStatusHelp

After importing the module you can run `Open-PSProjectStatusHelp` which will open a PDF version of this document in the default application associated with PDF files. Or if you are running PowerShell 7, you can use the `-AsMarkdown` dynamic parameter to read this file using markdown formatting. Not all markdown features may render properly in the console.

Class-Based

The project status is based on a private class-based definition. The PowerShell classes are used to construct the JSON file which in turn is used to create a `PSProject` object and update its properties.

```
Class PSProjectRemote {
  [string]$Name
  [string]$Url
  [gitMode]$Mode

  PSProjectRemote ($Name, $url, $mode) {
      $this.Name = $Name
      $this.url = $url
      $this.mode = $mode
  }
  #allow an empty remote setting
  PSProjectRemote() {
      $this.Name = ''
      $this.url = ''
  }
}


#I have formatted longer lines with artificial line breaks to fit a
#printed page.
Class PSProject {
```

3

```powershell
[string]$Name = (Split-Path (Get-Location).path -Leaf)
[string]$Path = (Convert-Path (Get-Location).path)
[DateTime]$LastUpdate = (Get-Date)
[string[]]$Tasks = @()
[PSProjectStatus]$Status = 'Development'
[Version]$ProjectVersion = (Test-ModuleManifest ".\$(Split-Path $pwd `
-Leaf).psd1" -ErrorAction SilentlyContinue).version
[string]$GitBranch = ''
#using .NET classes to ensure compatibility with non-Windows platforms
[string]$UpdateUser = "$([System.Environment]::UserDomainName)\`
$([System.Environment]::Username)"
[string]$Computername = [System.Environment]::MachineName
[PSProjectRemote[]]$RemoteRepository = @()
[string]$Comment = ''

[void]Save() {
    $json = Join-Path -Path $this.path -ChildPath psproject.json
    #convert the ProjectVersion to a string in the JSON file
    #convert the LastUpdate to a formatted date string
    $this | Select-Object @{Name = '$schema'; Expression = {
    'https://raw.githubusercontent.com/jdhitsolutions/PSProjectStatus/
    main/psproject.schema.json' } },
    Name, Path,
    @{Name = 'LastUpdate'; Expression = { '{0:o}' -f $_.LastUpdate }},
    @{Name = 'Status'; Expression = { $_.status.toString() }},
    @{Name = 'ProjectVersion'; Expression = {
      $_.ProjectVersion.toString()}},UpdateUser,Computername,
      RemoteRepository,Tasks,GitBranch,Comment |
    ConvertTo-Json | Out-File -FilePath $json -Encoding utf8
}
[void]RefreshProjectVersion() {
    $this.ProjectVersion = (Test-ModuleManifest ".\$(Split-Path $pwd `
     -Leaf).psd1" -ErrorAction SilentlyContinue).version
}
[void]RefreshUser() {
    $this.UpdateUser = "$([System.Environment]::UserDomainName)\`
    $([System.Environment]::Username)"
}
[void]RefreshComputer() {
    $this.Computername = [System.Environment]::MachineName
}
[void]RefreshRemoteRepository() {
    if (Test-Path .git) {
        $remotes = git remote -v
```

```powershell
            if ($remotes) {
                $repos = @()
                foreach ($remote in $remotes) {
                    $split = $remote.split()
                    $RemoteName = $split[0]
                    $Url = $split[1]
                    $Mode = $split[2].replace('(', '').Replace(')', '')
                    $repos += [PSProjectRemote]::new($RemoteName, $url,
                    $mode)
                } #foreach
                $this.RemoteRepository = $repos
            } #if remotes found
        }
    }

    [void]RefreshAll() {
        $this.RefreshProjectVersion()
        $this.RefreshUser()
        $this.RefreshComputer()
        $this.RefreshRemoteRepository()
        $this.Save()
    }
}```
```

The class includes a status enumeration.

```powershell
enum PSProjectStatus {
    Development
    Updating
    Stable
    AlphaTesting
    BetaTesting
    ReleaseCandidate
    Patching
    UnitTesting
    AcceptanceTesting
    Other
}
```

At this time it is not possible to include a user-defined project status. It is hoped that you can find something appropriate from the current status list.

The Age ScriptProperty and VersionInfo property sets are added to the object as type extensions.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Types>
  <Type>
    <Name>PSProject</Name>
    <Members>
      <PropertySet>
        <Name>versionInfo</Name>
        <ReferencedProperties>
          <Name>Name</Name>
          <Name>Status</Name>
          <Name>Version</Name>
          <Name>GitBranch</Name>
          <Name>LastUpdate</Name>
        </ReferencedProperties>
      </PropertySet>
      <AliasProperty>
        <Name>Version</Name>
        <ReferencedMemberName>ProjectVersion</ReferencedMemberName>
      </AliasProperty>
      <AliasProperty>
        <Name>Username</Name>
        <ReferencedMemberName>UpdateUser</ReferencedMemberName>
      </AliasProperty>
      <ScriptProperty>
        <Name>Age</Name>
        <GetScriptBlock> (Get-Date) - $this.lastUpdate </GetScriptBlock>
      </ScriptProperty>
    </Members>
  </Type>
</Types>
```

Note: Note that some screen shots may be incomplete as I am still adding properties to the PSProject class.

Creating a Project Status

To create a project status file, navigate to the module root and run New-PSProjectStatus. The default status is Development

Figure 1: New PSProject Status

You can update properties when you create the project status.

```
New-PSProjectStatus -LastUpdate (Get-Item .\*.psd1).LastWriteTime `
-Status Updating -tasks "update help"
```


Figure 2: new custom project status

The command will create `psproject.json` in the root folder.

```
{
  "$schema": "https://raw.githubusercontent.com/jdhitsolutions/
  PSProjectStatus/main/psproject.schema.json",
  "Name": "PSHelpDesk",
  "Path": "C:\\Scripts\\PSHelpDesk",
  "LastUpdate": "2024-02-20T09:47:33-05:00",
  "Status": "Updating",
  "ProjectVersion": "0.1.0",
  "UpdateUser": "PROSPERO\\Jeff",
  "Computername": "PROSPERO",
  "RemoteRepository": [],
  "Tasks": [
    "update help"
  ],
  "GitBranch": "dev",
  "Tags : [],
```

```
    "Comment": ""
}
```

Note that the update time is formatted as a UTC string. The project version will be pulled from the module manifest if found. You can set this to a different value manually in the JSON file or by running `Set-PSProjectStatus`.

Note: If you are using *git* with your module you may want to add `psproject.json` to your `.gitignore` file.

Getting a Project Status

The easiest way to view a project status is by using Get-PSProjectStatus.

```
PS C:\scripts\PSCalendar> Get-PSProjectStatus

   Name: PSCalendar [C:\Scripts\PSCalendar]

LastUpdate              Status        Tasks                    GitBranch        Age
----------              ------        -----                    ---------        ---
3/3/2024 10:24:49 AM    Patching      {Update help docu...        2.9.0    12.07:07
```

If the PowerShell host supports ANSI, a status of `Stable` will be displayed in Green. `Development` will be shown in Red and `Updating` in Yellow.

The module has a default list view.

```
PS C:\scripts\PSCalendar> Get-PSProjectStatus | Format-List

   Project: PSCalendar [C:\Scripts\PSCalendar]

Version    : 2.9.0
Status     : Patching
Tasks      : {Update help documentation, Issue #31,Issue #34,Issue #33}
GitBranch  : 2.9.0
LastUpdate : 3/3/2024 10:24:49 AM
```

This makes it easier to view tasks.

Updating a Project Status

To update the project status, you could always manually update the JSON file in your script editor. Use this code snippet to get the DateTime value in the proper format.

```
Get-Date -format o | Set-Clipboard
```

Paste the value into the file.

The Status value is an integer indicating a private enumeration value.

```
Development = 0
Updating = 1
Stable = 2
AlphaTesting = 3
BetaTesting = 4
ReleaseCandidate = 5
Patching = 6
UnitTesting = 7
AcceptanceTesting = 8
Other = 9
Archive = 10
```

Or use the Set-PSProjectStatus function.

```
PS C:\scripts\PSHelpDesk\> $tasks = "add printer status function",
"revise user password function"
PS C:\scripts\PSHelpDesk> Set-PSProjectStatus -LastUpdate (Get-Date) `
-Status Development -Tasks $tasks -Concatenate


   Name: PSHelpDesk [C:\Scripts\PSHelpDesk]

LastUpdate              Status       Tasks                 GitBranch        Age
----------              ------       -----                 ---------        ---
3/15/2024 5:53:54 PM    Development  {update help, add...        dev    00.00:00
```

When defining tasks, use -Concatenate to append the tasks. Otherwise, tasks will be overwritten with the new value.

Source Control Status

The commands in this module assume you are most likely using git for source control. The status object will automatically detect the local git branch. It will also detect the primary remote repositories.

Figure 3: remote repository status

Manually Updating with the Object

The PSProject class has been updated since the first version of this module was released. You can use the object's methods to refresh some properties. Here is an example of an incomplete status.

```
PS C:\Scripts\WingetTools> Get-PSProjectStatus | Select-Object *

Name            : WingetTools
Status          : Stable
Version         :
GitBranch       : main
LastUpdate      : 3/17/2024 9:46:35 AM
Age             : 9.00:22:39.3936893
Path            : C:\Scripts\WingetTools
ProjectVersion  :
UpdateUser      : THINKX1-JH\Jeff
Computername    :
RemoteRepository : {}
Tasks           : {}
Comment         :
Tags            : {}
```

To update, get a reference to the project status object.

```
$p = Get-PSProjectStatus
```

Get-Member will show you the available methods.

Figure 4: psproject methods

Invoke the methods that apply to your project. You need to invoke the `Save()` method to commit the changes to the JSON file.

```
$p.RefreshComputer()
$p.RefreshUser()
$p.RefreshProjectVersion()
$p.RefreshRemoteRepository()
$p.save()
```

As an alternative can use the `RefreshAll()` method which will invoke all the refresh methods **and** save the file.

Project Tasks

This module is intended to be a *simple* project management tool. You can use it to track tasks or to-do items. These are added to the `Tasks` property as an array of strings. You can manually add them to the JSON file or use the `Set-PSProjectStatus` function.

```
C:\Scripts\PSProjectStatus> $params = @{
 Tasks="Update missing online help links"
 Concatenate=$true
 }

C:\Scripts\PSProjectStatus> Set-PSProjectStatus @params

   Name: PSProjectStatus [C:\Scripts\PSProjectStatus]
```

LastUpdate	Status	Tasks	GitBranch	Age
12/22/2024 9:08:30 AM	Updating	{Consider a schema …	0.11.0	00.00:00

Or you can use the task-related commands.

```
PS C:\Scripts\PSProjectStatus> Get-PSProjectTask

   Name: PSProjectStatus [C:\Scripts\PSProjectStatus]

   ● Consider a schema update for tasks [1]
   ● Create TUI-based management tools [2]
   ● Add support for tags Issue 8. This would need a schema update. [3]
   ● Update README [4]
   ● Pester tests [5]
   ● Update missing online help links [6]
```

Figure 5: Get-PSProjectTask

If the PowerShell host supports it, you should get ANSI formatting. The task ID is automatically generated for each item and displayed in square brackets.

You can also add a task.

```
PS C:\Scripts\PSProjectStatus> New-PSProjectTask -TaskDescription "Add localized string data" -PassTh
ru

   Name: PSProjectStatus [C:\Scripts\PSProjectStatus]

   ● Add localized string data [7]
```

Figure 6: New-PSProjectTask

You can manually remove items from the JSON file or use the Remove-PSProjectTask function. You will need to know the task id.

```
Remove-PSProjectTask -TaskID 4
```

Note: *The* PSProjectTask *object is defined in a PowerShell class. The class is defined with future enhancements in mind. Not all defined properties are used at this time.*

Project Management

If you have many projects, you can use this module to manage all of them.

```powershell
Get-ChildItem -Path c:\scripts -Directory |
Get-PSProjectStatus -WarningAction SilentlyContinue
```


Figure 7: list projects

You will want to suppress Warning messages. If you are running PowerShell 7 and have the `Microsoft.PowerShell.ConsoleGuiTools` module installed, you can run a script like this:

```powershell
#requires -version 7.2
#requires -module Microsoft.PowerShell.ConsoleGuiTools

#open a project using the PSProject status

Import-Module PSProjectStatus -Force

#Enumerate all directories and get the project status for each
$all = Get-ChildItem -Path C:\scripts -Directory |
Get-PSProjectStatus -WarningAction SilentlyContinue

#Pipe directory output to Out-ConsoleGridView
#and open the selected project in VS Code
$all | Sort-Object Status, LastUpdate |
Select-Object Path, Status,
@{Name = "Tasks"; Expression = { $_.Tasks -join ',' } },
GitBranch, LastUpdate |
```

13

```
Out-ConsoleGridView -Title "PSProject Management" -OutputMode Single |
ForEach-Object { code $_.path }
```

This will give you a list of projects.

Figure 8: project list

You can select a single project, press Enter, and open the folder in VS Code. You could write a similar script for Windows PowerShell using `Out-GridView`.

Get-PSProjectReport

Beginning with version `0.10.0` you can use `Get-PSProjectReport` to simplify project management.

You can get all of your projects.

```
Get-PSProjectReport c:\scripts
```

You can filter by status.

```
PS C:\> Get-PSProjectReport c:\scripts -Status Other


   Name: PSMessaging [C:\Scripts\PSMessaging]
```

LastUpdate	Status	Tasks	GitBranch	Age

```
----------          ------          -----          ----------          ---
7/20/2022 11:58:54 AM  Other                   {}                       master  192.02:11
```

And you can filter by age.

```
PS C:\> Get-PSProjectReport c:\scripts -NewerThan 10 -Status Stable


   Name: PluralsightTools [C:\Scripts\PluralsightTools]

LastUpdate              Status              Tasks              GitBranch              Age
----------              ------              -----              ---------              ---
1/20/2023 2:20:39 PM    Stable              {convert modu ...       main    07.23:51
```

Project Tags

Support for tags was added in version 0.12.0. You can define tags when you create the project status file.

```
New-PSProjectStatus -Tasks "prototype" -Tags tui - -version 0.2.0
```

Or you can add them later.

```
Set-PSProjectStatus -Tags "beta","tui"
```

When using this command you need to redefine existing tags. Or add the tags manually to the JSON file.

You can view tags with a formatted list view.

```
PS C:\work\terminalgui> Get-PSProjectStatus | Format-List


   Project: terminalgui [C:\work\terminalgui]

Version    : 0.2.0
Status     : Development
Tasks      : {prototype}
Tags       : {beta, tui}
GitBranch  :
LastUpdate : 12/27/2024 5:11:30 PM
Age        : 00:02:48.0251636
```

You are most likely to use tags when managing multiple projects. Get-PSProjectReport includes a -Tag parameter so that you can filter from your parent folder.

```
PS C:\> Get-PSProjectReport -path c:\scripts -Tag json

   Name: PSProjectStatus [C:\Scripts\PSProjectStatus]

LastUpdate              Status        Tasks                 GitBranch        Age
----------              ------        -----                 ---------        ---
12/27/2024 5:16:52 PM   Updating      {Create TUI-based m…     0.12.0   00.00:00
```

If you want to remove tags, either manually edit the JSON file or use `Set-PSProjectStatus` and set an empty array.

```
Set-PSProjectStatus -Tags @()
```

Removing Project Status

If no you longer want to track the project status for a given folder, simply delete the associated JSON file. As an alternative, you can set the status to `Archive`.

Module Extensions

Type Extensions

The commands in this module have defined type extensions. Alias and script properties have been defined.

```
PS C:\Scripts\PSProjectStatus> Get-PSProjectstatus |
Get-Member -MemberType Properties,PropertySet

   TypeName: PSProject

Name              MemberType     Definition
----              ----------     ----------
Username          AliasProperty  Username = UpdateUser
Version           AliasProperty  Version = ProjectVersion
Comment           Property       string Comment {get;set;}
Computername      Property       string Computername {get;set;}
GitBranch         Property       string GitBranch {get;set;}
LastUpdate        Property       datetime LastUpdate {get;set;}
Name              Property       string Name {get;set;}
Path              Property       string Path {get;set;}
ProjectVersion    Property       version ProjectVersion {get;set;}
RemoteRepository  Property       PSProjectRemote[] RemoteRepository   ...
Status            Property       PSProjectStatus Status {get;set;}
```

```
Tags             Property       string[] Tags {get;set;}
Tasks            Property       string[] Tasks {get;set;}
UpdateUser       Property       string UpdateUser {get;set;}
Info             PropertySet    Info {Name, Status, Version, GitBranc...
versionInfo      PropertySet    versionInfo {Name, Status, Version, G...
Age              ScriptProperty System.Object Age {get=(Get-Date) -   ...
```

The property sets make it easier to display a group of related properties.

```
PS C:\Scripts\PSProjectStatus> Get-PSProjectstatus | Select-Object Info


Name      : PSProjectStatus
Status    : AcceptanceTesting
Version   : 0.13.0
GitBranch : 0.13.0
Tasks     : {Create TUI-based management tools, Consider extending schema
            for a structured Task item [Issue 10],
            Pester tests}
Tags      : {}
Comment   : none


PS C:\Scripts\PSProjectStatus> Get-PSProjectStatus |
Select-Object VersionInfo,Age


Name       : PSProjectStatus
Status     : AcceptanceTesting
Version    : 0.13.0
GitBranch  : 0.13.0
LastUpdate : 12/30/2023 1:43:37 PM
Age        : 00:03:56.0703713
```

Formatting

The module uses custom and default formatting for projects and tasks. The default format is a table. There are examples you can see in several screenshots above. You can use also `Format-List`.

```
PS C:\Scripts\PSProjectStatus> Get-PSProjectStatus | Format-List


   Project: PSProjectStatus [C:\Scripts\PSProjectStatus]


Version   : 0.15.0
Status    : Updating
Tasks     : {Create TUI-based management tools, Consider extending schema for
```

```
              a structured Task item [Issue 10], Pester tests, Consider adding a
              project type, eg module, to the schema…}
Tags         : {json, class-based}
GitBranch    : 0.15.0
LastUpdate   : 7/16/2024 1:07:22 PM
Age          : 173.20:28:04
```

There is also a named view you can use.

```
PS C:\Scripts\PSProjectStatus> Get-PSProjectStatus |
Format-List -View info


   Project: PSProjectStatus [C:\Scripts\PSProjectStatus]


Status  : Updating
Tasks   : {Create TUI-based management tools, Consider extending schema
          for a structured Task item [Issue 10], Pester tests, Consider
          adding a project type, eg module, to the schema…}
Tags    : {json, class-based}
Comment :
Age     : 173.20:28:37
```

Verbose, Warning, and Debug

The commands in this module use localized string data to display verbose, warning, and debug messages. The module uses a private helper function to display verbose messaging. Each module command can be identified with a different ANSI color scheme.

Figure 9: Sample verbose output

Note *Localized string data to languages other than English was done with*

GitHub CoPilot, so I can't guarantee the accuracy or quality of the translations. As of version `0.16.0` *the supported cultures are* `fr-FR`

The defined ANSI sequences are stored in a hashtable variable called `$PSProjectANSI`.

```
$PSProjectANSI = @{
    'Get-PSProjectGitStatus' = '[1;38;5;51m'
    'Get-PSProjectReport'    = '[1;38;5;111m'
    'Get-PSProjectStatus'    = '[1;96m'
    'Get-PSProjectTask'      = '[1;38;5;10m'
    'New-PSProjectStatus'    = '[1;38;5;208m'
    'New-PSProjectTask'      = '[1;38;5;159m'
    'Remove-PSProjectTask'   = '[1;38;5;195m'
    'Set-PSProjectStatus'    = '[1;38;5;214m'
    Default                  = '[1;38;5;51m'
}
```

You can change a setting by modifying the variable. You can use ANSI sequences or `$PSStyle`

```
$PSProjectANSI["Get-PSProjectStatus"] = "[1;92m"
$PSProjectANSI["Get-PSProjectGitStatus"] = $PSStyle.Foreground.Cyan
```

These changes only persist for the duration of your PowerShell session or until you re-import the module. Use your profile script to import the module and update the variable.

```
Import-Module PSProjectStatus
$PSProjectANSI["Get-PSProjectStatus"] = "[1;38;5;140m"
$PSProjectANSI["Get-PSProjectGitStatus"] = "[1;38;5;77m"
```

! You must use a PowerShell console that supports ANSI escape sequences. The PowerShell ISE **does not** support this feature.

Editor Integration

If you import this module into your PowerShell editor, either Visual Studio Code or the PowerShell ISE, the module will add an update function called `Update-PSProjectStatus`. You can run the command from the integrated terminal or use the appropriate shortcut (see below). The command will the status based on user input, update the `LastUpdate` time to the current date and time, update the project version from the module manifest (if found), and update the git branch if found.

You need to make sure your terminal or console window is set to your project's

root directory.

PowerShell ISE

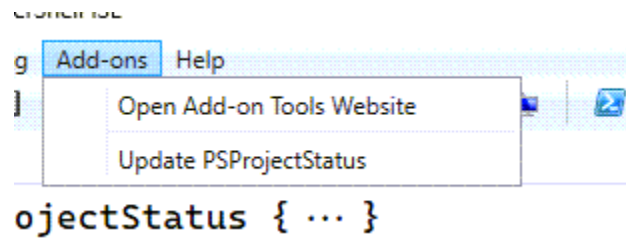If you import the module in the PowerShell ISE, it will add a menu shortcut under `Add-Ons`.

Figure 10: add-on menu

Click the shortcut and a status menu will be displayed in the console pane.

```
PS C:\scripts\PSProjectStatus> Update-PSProjectStatus

    Status Options
    ---------------

    [1] Development        [6] ReleaseCandidate
    [2] Updating           [7] Patching
    [3] Stable             [8] UnitTesting
    [4] AlphaTesting       [9] AcceptanceTesting
    [5] BetaTesting        [10] Other

Select a project status. Enter no value to cancel: 2


Name        : PSProjectStatus
Status      : Updating
Version     : 0.6.0
GitBranch   : 0.6.0
LastUpdate  : 3/29/2022 2:13:03 PM



PS C:\scripts\PSProjectStatus>
```

Figure 11: ISE update status

Select a status and press Enter The function will call `Set-PSProjectStatus` and display the updated `versioninfo` property.

VS Code

Likewise, in VS Code open the command palette and go to `PowerShell: Show Additional commands from PowerShell modules`. You should see an option to update.
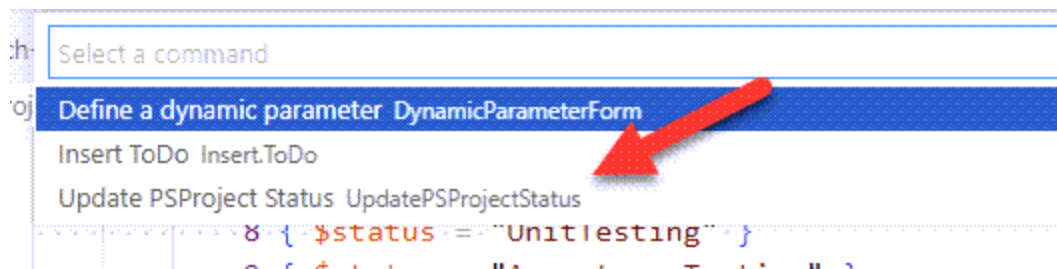
21

Figure 12: VSCode additional command

Select the menu choice and switch to the integrated terminal window.
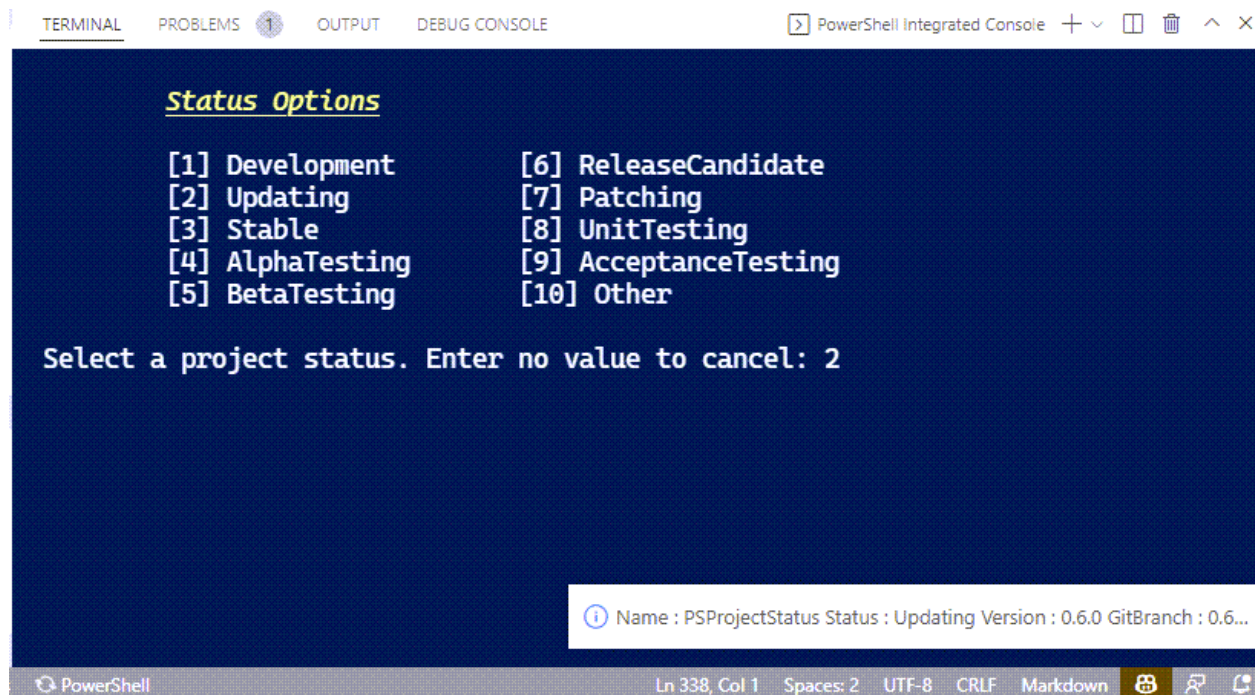

Figure 13: VSCode update status

The menu will loop and display until you enter a valid number or press Enter with no value. The summary will be displayed as a VSCode information message.

JSON Schema

A public JSON schema file was published with `v0.8.0`. If you edit the `psproject.json` file in VSCode, you should get tab completion for many of the settings. If you have a configuration file created with an earlier version of the module, run `Set-PSProjectStatus` with any parameter. This will insert the schema reference into the JSON file. Then you can edit the file in VSCode.

Cross-Platform Support

The commands in this module should work under PowerShell 7.x cross-platform. Beginning with version 0.14.0, commands have been updated to store the path using operating system-appropriate paths. The only potential issue you might encounter is if you manage the same project files in Windows and Linux, e.g. using WSL. If that is the case, I recommend you run `Set-PSProjectStatus` before running any other commands. This will ensure the path in the JSON file is correct.

Road Map

These are a few things I'm considering or have been suggested.

- Additional properties
  - priority
  - project type
- Editor integration to manage project tasks
- Extending the schema to support tasks
- Archiving completed tasks to a separate JSON file
- A WPF or TUI form to display the project status and make it easier to edit tasks

💬 If you have any suggestions on how to extend this module or tips to others on how you are using it, please feel free to use the Discussions section of this module's GitHub repository.

Note: 👍 Project icon by Icons8