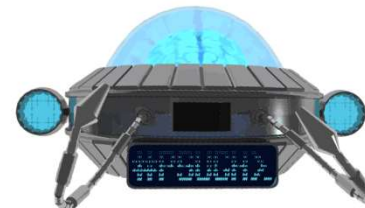


MSGraph PowerShell module

Deep dive into a comfortable implementation of MS Graph API

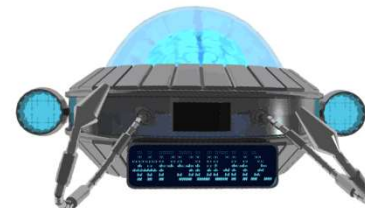


PS > about_Speaker

- Andreas Bellstedt
 - IT infrastructure guy and PowerShell fellow

 |  @AndiBellstedt

 github.com/AndiBellstedt



PS > Agenda

- History – The story behind the module
- Architecture – Bricks and concrete to build it solid
- CI/CD – The way it is released

A detailed illustration of a magical forest. A glowing blue figure with a radiant, starburst-like headpiece stands in the center-right, holding a small glowing orb. A beam of light emanates from the figure, shining down onto a small stream that flows through the forest floor. The trees are tall and slender, with thick, gnarled roots. The lighting is soft and ethereal, with a mix of green and blue hues. The overall atmosphere is mystical and enchanting.

The story behind the module
Once upon a time...

History

- There was a lonely department, named accounting
- They were trying to invent some kind of digitalization process
- Exchange was migrated to the cloud, but the accounting system remain onprem
- Some brave IT guys came along, talk to the accounting department and decide going into an adventure to rescue all the lonely billing mails and their attachments
- As they can use PowerShell, the story comes like this...
 - Solving a single problem was not enough
 - A platform for further potential challenges should be created
- A new module was born - **MSGraph**



PowerShell MSGraph

Interacting with
Microsoft Graph

MSGraph - Interacting with Microsoft Graph

| Platform | Information |
|--------------------|--|
| PowerShell gallery | psgallery v1.3.0 platform windows downloads 3.1k |
| GitHub | release v1.3.0 license MIT open issues 0 last commit: master march last commit: development march |

The MSGraph module is a wrapper around the Graph API of Microsoft. It offers tools to interact with exchange online (more services planned and seamlessly supportable).

All cmdlets are build with

- powershell regular verbs
- mostly with pipeling availabilities
- comprehensive logging on verbose and debug channel

Note: Project is still in its infancy, more to come

Installation

Install the module from the PowerShell Gallery (systemwide):

```
Install-Module MSGraph
```

or install it only for your user:

```
Install-Module MSGraph -Scope CurrentUser
```

PS > Demo



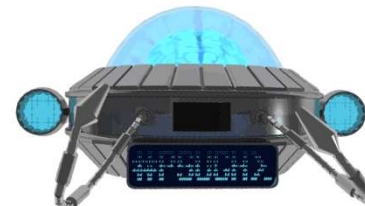
- Enough fairy tale... let's get technical





Architecting a module

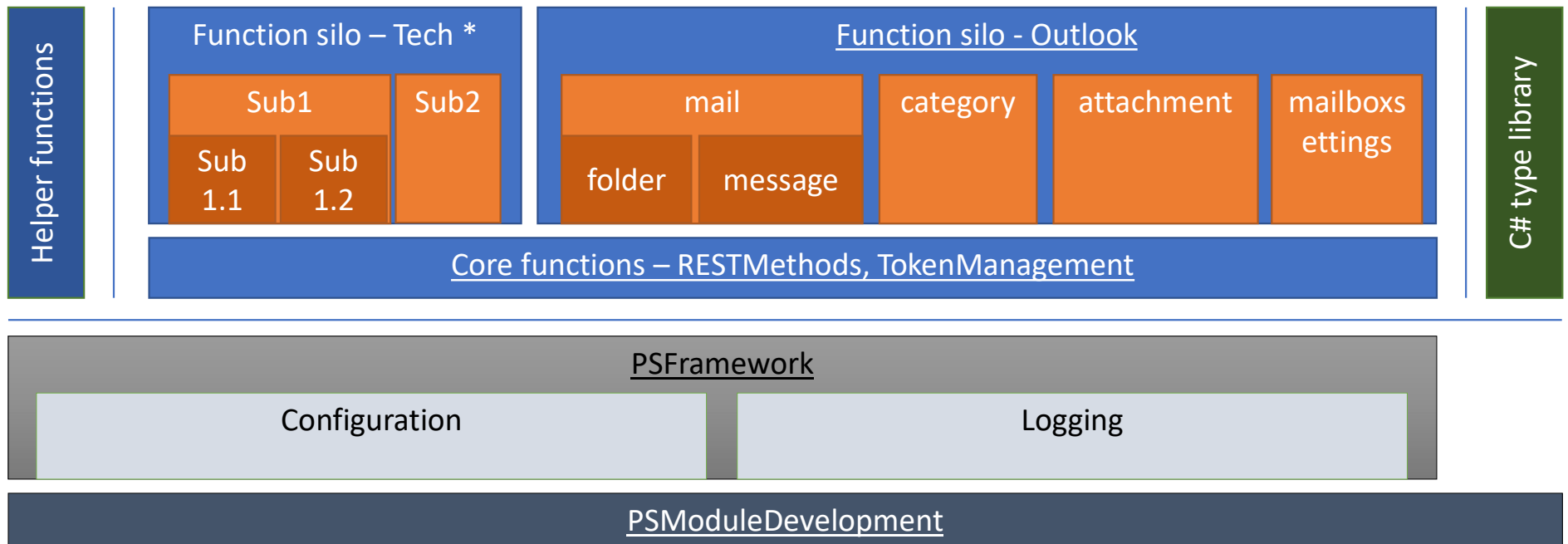
Bricks and concrete to build it solid



PS > Expectation on the module

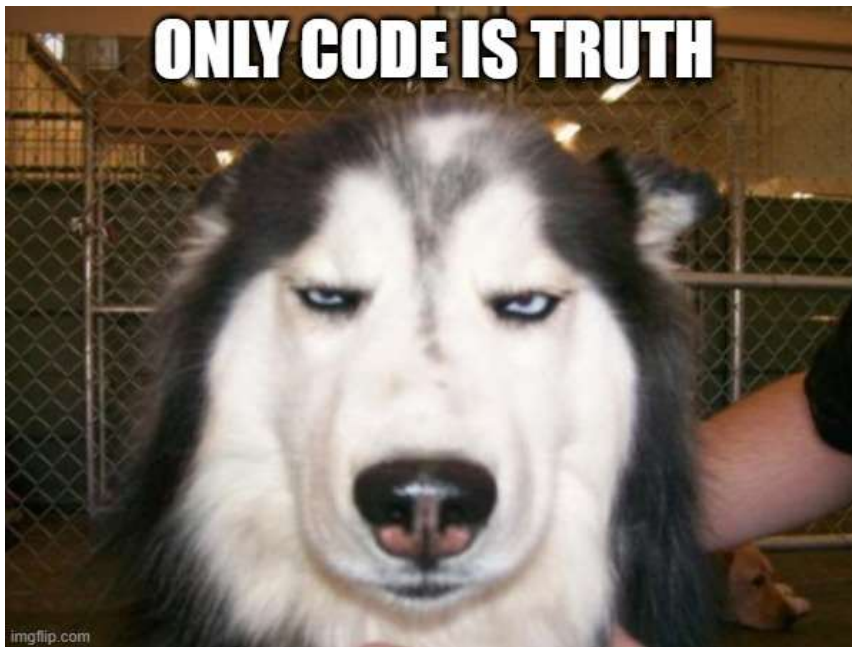
- Maintainable & scalable module structure
- Practical & comfortable usage
 - Documentation and help for every function
 - Functions support PS pipeline
- Comprehensible Configuration
 - Customizable
 - Configurable
- Detailed logging
- Reliable naming schema

PS > Architectural landscape

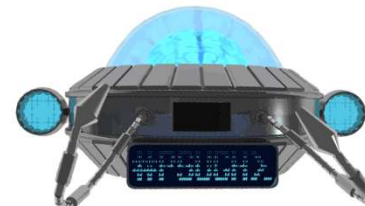


<https://learn.microsoft.com/en-us/graph/auth/auth-concepts?view=graph-rest-1.0>

PS > Demo



Digging inside the module



PS > Naming schema

- Avoid collisions on function names with other modules
- Prefix module commands
 - Mga
- Descriptive commands
 - MgaMailMessage,
MgaMailFolder,
...
- Commonly used verbs
 - Get, Set, Add,
Invoke, Remove,
...
- Aliases for practical usage
 - Connect <-> New-MgaAccessToken
 - Export <-> Save
 - Update <-> Set

| Noun without Prefix | Verb | Name | Type | Resolved command |
|---------------------|----------|------------------------------|----------|------------------------------|
| AccessToken | New | New-MgaAccessToken | Function | |
| AccessToken | Register | Register-MgaAccessToken | Function | |
| AccessToken | Update | Update-MgaAccessToken | Function | |
| AccessToken | Get | Get-MgaAccessTokenRegistered | Function | |
| AccessToken | Get | Get-MgaRegisteredAccessToken | Alias | Get-MgaAccessTokenRegistered |
| MailAttachment | Add | Add-MgaMailAttachment | Function | |
| MailAttachment | Export | Export-MgaMailAttachment | Function | |
| MailAttachment | Get | Get-MgaMailAttachment | Function | |
| MailAttachment | Remove | Remove-MgaMailAttachment | Function | |
| MailAttachment | Save | Save-MgaMailAttachment | Alias | Export-MgaMailAttachment |
| MailFolder | Get | Get-MgaMailFolder | Function | |
| MailFolder | Move | Move-MgaMailFolder | Function | |
| MailFolder | New | New-MgaMailFolder | Function | |
| MailFolder | Remove | Remove-MgaMailFolder | Function | |
| MailFolder | Rename | Rename-MgaMailFolder | Function | |

Infrastructure requirements

Client

- PowerShell module MSGraph
- Module dependencies
 - PSFramework

Server

- **Entra ID** tenant / Microsoft Account
- Registered Application
- Permissions
 - App registration
 - App granting

The screenshot shows the 'PowerShell.MSGraph | Branding' configuration page. The left sidebar contains a navigation menu with 'Manage' expanded, showing options like Overview, Quickstart, Integration assistant, Branding (selected), Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, and Owners. The main content area has a top bar with a search box, 'Save', 'Discard', and 'Got feedback?' buttons. Below this, the 'Name' field is set to 'PowerShell.MSGraph'. The 'Logo' field shows a custom logo with 'PS' and colored dots. The 'Upload new logo' field has a 'Select a file' button. The 'Home page URL' is 'https://github.com/AndiBelstedt/MSGraph'. The 'Terms of service URL' is 'e.g. https://example.com/termsofservice'. The 'Privacy statement URL' is 'e.g. https://example.com/privacystatement'. The 'Publisher domain' is 'Unverified' with a warning icon and a 'Configure a domain' link.

^ Essentials

Display name : PowerShell.MSGraph

Application (client) ID : 5e79add2-6288-4d91-bebc-cae920227404

Object ID : 9d831c88-73a6-4456-baca-9681dea296e0

What is CI/CD



Continuous integration - Wikipedia

In [software engineering](#), continuous integration (CI) is the practice of merging all developers' working copies to a shared [mainline](#) several times a day.

CI is intended to be used in combination with automated unit tests written through the practices of [test-driven development](#).

- Maintain a code repository
- Automate the build
- Make the build self-testing

Continuous deployment - Wikipedia

Continuous deployment (CD) is a [software engineering approach](#) in which software functionalities are delivered frequently through automated [deployments](#). CD contrasts with [continuous delivery](#), a similar approach in which software functionalities are also frequently delivered and deemed to be potentially capable of being deployed but are actually not deployed.

- Automate the build
- Release after successful build and test

PS > The way it is released



- Continuous integration & Continuous deployment
- At least 2 branches
 - Master & Development
- Branch protection rules
- The real magic is in:



Azure Pipelines

Protect matching branches

☐ Require pull request reviews before merging
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

☒ Require status checks to pass before merging
Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☒ Require branches to be up to date before merging
This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

Status checks found in the last week for this repository

Branch protection rules Add rule

Define branch protection rules to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to branch protection rules? [Learn more](#).

| | | | |
|-------------|-------------------------------|------|--------|
| master | Currently applies to 1 branch | Edit | Delete |
| Development | Currently applies to 1 branch | Edit | Delete |

PS > Azure DevOps - Pipelines



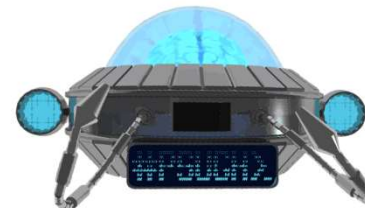
- Defined in yaml files
 - Included inside git repo of module
- Pipeline: Validate
 - Triggered by every commit in Development
 - Triggered by pull request on Master & Development branch
- Pipeline: Build & publish
 - Triggered by commit on master

```
1 pool:
2   - vmImage: 'windows-latest'
3
4   # Continuous integration only on branch Development
5   trigger:
6     branches:
7       - include:
8         - Development
9
10  # Pull request validation only on branch master & development
11  pr:
12    branches:
13      - include:
14        - master
15        - Development
16
17  steps:
18    - task: PowerShell@2
19      displayName: Ensure prerequisites
20      inputs:
21        targetType: filePath
22        filePath: './build/vsts-prerequisites.ps1'
23        arguments: '-ModuleName $(system.teamProject)'
24
25    - task: PowerShell@2
26      displayName: Validate code compliance
27      inputs:
28        targetType: filePath
29        filePath: './build/vsts-validate.ps1'
30        arguments: '-ModuleName $(system.teamProject)'
31
32    - task: PublishTestResults@2
33      displayName: 'Publish Test Results **/TEST-*.xml'
34      inputs:
35        testResultsFormat: NUnit
36        condition: always()
```

PS > Demo



- GitHub
- Azure DevOps



PS > Conclusion

- Projects from curiosity don't go the easy way
- Curiosity brings knowledge
- Module is built with
 - Scaling structure in mind
 - Verbose logging to tell what is going on under the hood
- Module MSGraph is on the PowerShell Gallery
 - (Currently) singularly handling Outlook mail items
 - (Maybe) growing feature set in the future

PS > Question & Answers



– Thank you –

Andreas Bellstedt

 |  @AndiBellstedt

 github.com/AndiBellstedt