



PowerShell Conference Europe

No more module not found

Stephan van Rooij



Starting at 13:00 CEST (UTC +2)

Many thanks to our sponsors:

Devolutions



Jane Street



Microsoft



SynEdgy

No More, Module not found

Session about AssemblyLoadContext 🤔



Stephan van Rooij

Professional burglar
Software architect
Microsoft MVP
Speaker



You might know me from



<https://linkedin.com/in/stephanvanrooij>

Stephan van Rooij ✓

Software Architect @ Smartersoft B.V. | Hacker | Public speaker |
Microsoft MVP Security & .NET

Vught, North Brabant, Netherlands · [Contact info](#)

5,246 followers · 500+ connections

Open to

Add section

Visit my website

Resources



Smartersoft B.V.



Fontys Hogescholen

You might know me from

Coding Stephan 

Projects Search    

**Stephan van Rooij**
Software developer specialized in API security. Microsoft MVP in Security and Graph since October 2022 and public speaker.

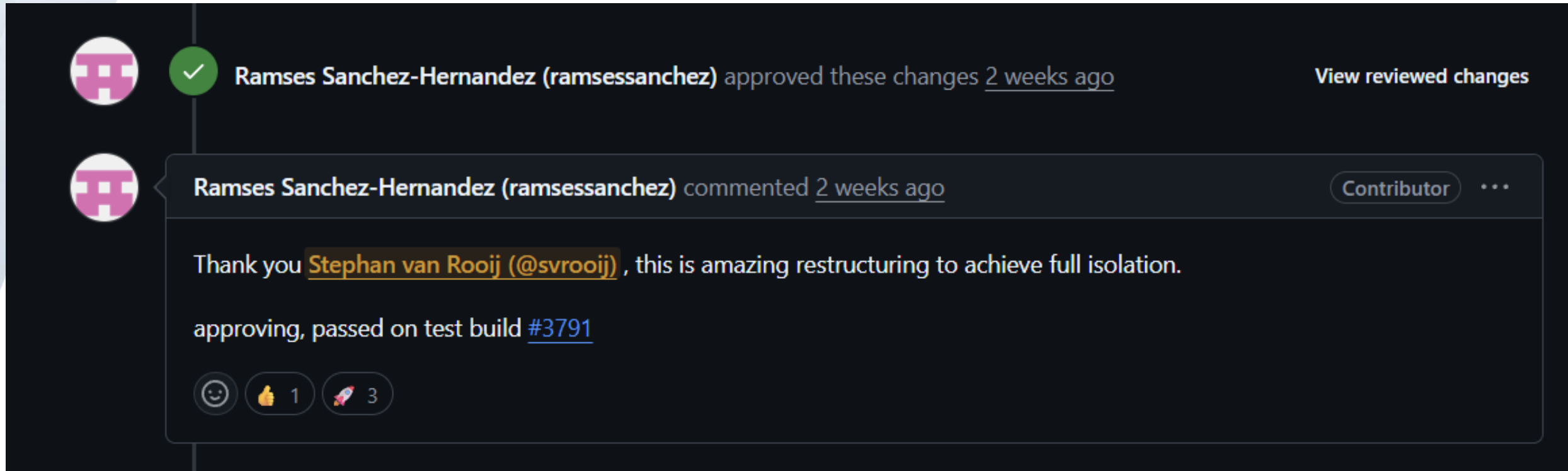
<https://svrooij.io>

Microsoft Graph Dependency issues
Oct 6, 2026

Hacking at Scottish Summit
Oct 3, 2026

Fido2 showdown: Security key form factors
Sep 28, 2026

Or from #3789



The image shows a screenshot of a GitHub pull request interface. At the top, a green checkmark icon is next to the text "Ramses Sanchez-Hernandez (ramsessanchez) approved these changes 2 weeks ago". To the right of this is a link "View reviewed changes". Below this, a comment by the same user is shown. The comment text is "Thank you [Stephan van Rooij \(@svrooij\)](#), this is amazing restructuring to achieve full isolation. approving, passed on test build [#3791](#)". At the bottom of the comment are three reaction icons: a smiley face, a thumbs up (with a count of 1), and a rocket (with a count of 3). The GitHub logo is visible on the left side of the interface.

  Ramses Sanchez-Hernandez (ramsessanchez) approved these changes [2 weeks ago](#) [View reviewed changes](#)

 Ramses Sanchez-Hernandez (ramsessanchez) commented [2 weeks ago](#) [Contributor](#) [...](#)

Thank you [Stephan van Rooij \(@svrooij\)](#), this is amazing restructuring to achieve full isolation.
approving, passed on test build [#3791](#)

  1  3

<https://github.com/microsoftgraph/msgraph-sdk-powershell/pull/3789>

Polls



I have some polls, use the Emote button to respond




```
PS C:\Users\StephanvanRoij> Connect-ExchangeOnline
Error Acquiring Token:
System.NullReferenceException: Object reference not set to an instance of an object.
   at Microsoft.Identity.Client.Platforms.Features.RuntimeBroker.RuntimeBroker..ctor(CoreUIParent uiParent, ApplicationC
onfiguration appConfig, ILoggerAdapter logger)
   at Microsoft.Identity.Client.Broker.BrokerExtension.<>c.<AddRuntimeSupport>b__3_0(CoreUIParent uiParent, ApplicationC
onfiguration appConfig, ILoggerAdapter logger)
   at Microsoft.Identity.Client.PlatformsCommon.Shared.AbstractPlatformProxy.CreateBroker(ApplicationConfiguration appCo
nfig, CoreUIParent uiParent)
   at Microsoft.Identity.Client.Internal.Requests.InteractiveRequest.ExecuteAsync(CancellationToken cancellationTok
en)
   at Microsoft.Identity.Client.Internal.Requests.InteractiveRequest.ExecuteAsync(CancellationToken cancellationTok
en)
--- End of stack trace from previous location ---
   at Microsoft.Identity.Client.Utils.StopwatchService.MeasureCodeBlockAsync(Func`1 codeBlock)
   at Microsoft.Identity.Client.Internal.Requests.RequestBase.RunAsync(CancellationToken cancellationToken)
   at Microsoft.Identity.Client.ApiConfig.Executors.PublicClientExecutor.ExecuteAsync(AcquireTokenCommonParameters commo
nParameters, AcquireTokenInteractiveParameters interactiveParameters, CancellationToken cancellationToken)
   at Microsoft.Exchange.Management.AdminApiProvider.Authentication.MSALTokenProvider.GetAccessTokenAsync(String claims,
String cmdletId)
OperationStopped: Object reference not set to an instance of an object.
```

Assembly Load Context

Assembly Load Context

- PowerShell 6+
- Responsible for loading other dlls
- One **Shared** Assembly Context
- Lookup by name
(and minimum version)

Issues with Assembly Load Context

- higher version $\neq$ compatible
- Module code cannot be certain of specific version

Custom Assembly Load Context

1. Create **IModuleAssemblyInitializer**
2. Create custom **AssemblyLoadContext**
3. Connect custom loader to **AssemblyLoadContext.Default.Resolving**

Is it that easy?

100 for yes



for NO!



for
tell me more

Some rules

1.

**Commandlets go in
Default Assembly context!**

2.

**Commandlets can only output
PowerShell types**

2.

Commandlets can only output PowerShell types



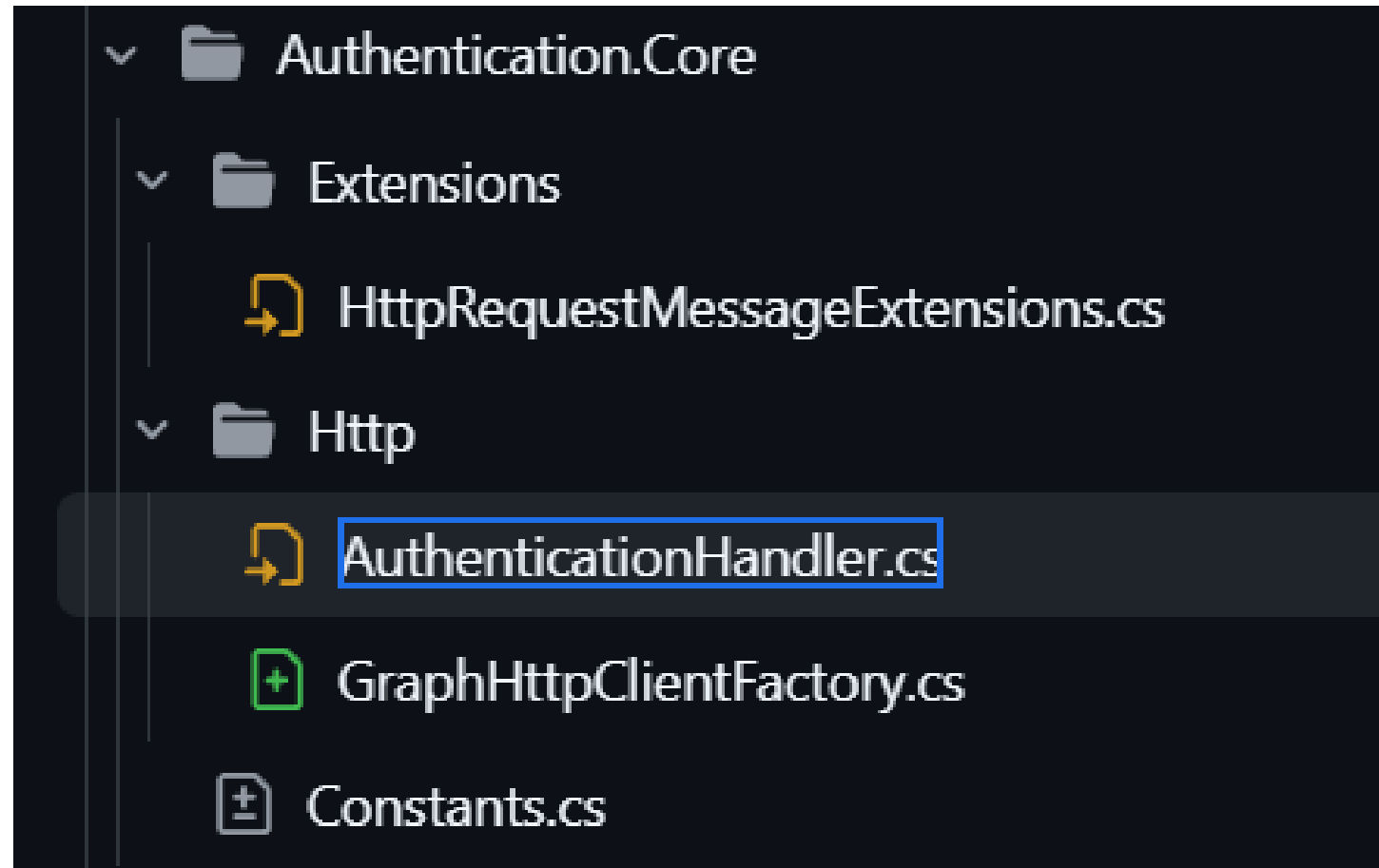
```
switch (OutputType)
{
    case OutputType.HashTable:
        responseString = await response.Content.ReadAsStringAsync();
        var hashTable = responseString.ConvertFromJson(true, null, out error);
        ThrowIfError(error);
        WriteObject(hashTable);
        break;
    case OutputType.PSObject:
        responseString = await response.Content.ReadAsStringAsync();
        var psObject = responseString.ConvertFromJson(false, null, out error);
        ThrowIfError(error);
        WriteObject(psObject, true);
        break;
    case OutputType.HttpResponseMessage:
        WriteObject(response);
}
```

3.

**External dependencies
go in a separate assembly**

3.

External dependencies go in a separate assembly



4.

You cannot call the
loader directly

You cannot call the **loader** directly



```
private static Type InitializeIsolatedLoadContext()
{
    string loaderPath = Path.Combine(s_psEditionDependencyFolder, LoaderAssemblyFileName);
    if (!File.Exists(loaderPath))
    {
        throw new FileNotFoundException(
            $"The Microsoft.Graph.Authentication module is incomplete: '{loaderPath}' was not found.", loaderPath);
    }

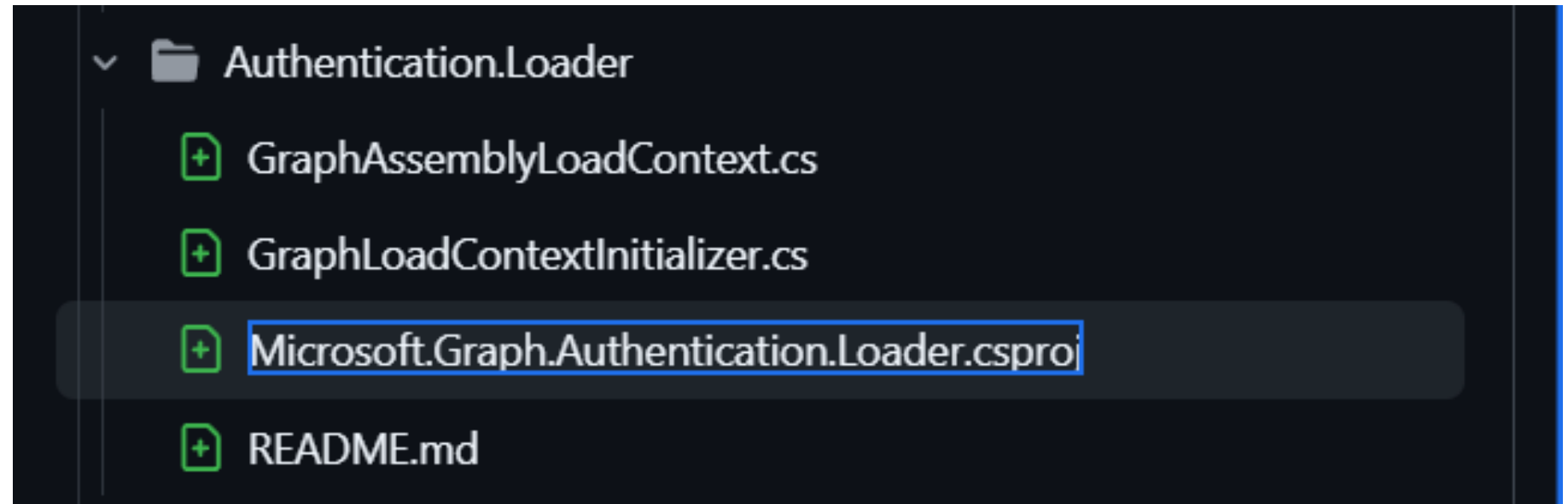
    Assembly loader = Assembly.LoadFrom(loaderPath);
    Type initializer = loader.GetType(LoaderInitializerTypeName, throwOnError: true);
    MethodInfo initialize = initializer.GetMethod("Initialize", BindingFlags.Public | BindingFlags.Static, null, new[] { type
    initialize.Invoke(null, new object[] { s_dependencyFolder, s_psEditionDependencyFolder });
    return initializer;
}
```

5.

**The loader must be
a separate assembly**

The loader must be a separate assembly

5.



6.

**Setup how module
is imported**

Code



6.

Setup how module is imported



1. Microsoft.Graph.Authentication.Loader is loaded and initializes private load context
2. Microsoft.Graph.Authentication is imported
3. Microsoft.Graph.Authentication.Core loads into **private context** when needed
4. When needed **Core dependencies** are loaded into the **private context**
5. Newtonsoft.Json is always loaded from **default context**

Microsoft.Graph.Authentication.Loader



main ▾

msgraph-sdk-powershell / src / Authentication / Authentication / Microsoft.Graph.Authentication.psm1



svrooij Completely isolating Microsoft.Graph.Authentication to never have dep...  

Code

Blame

49 lines (44 loc) · 2.47 KB 

```
1
2 # On PowerShell 7+ set up the isolated AssemblyLoadContext *before* the module dll is loaded, so that
3 # Microsoft.Graph.Authentication.Core and its dependencies (Azure.Identity, MSAL, Kiota, ...) never bind into the
4 # default load context and cannot conflict with other modules. Windows PowerShell 5.1 has no load contexts, so there the
5 # dependencies are pre-loaded with Assembly.LoadFrom (the ModuleInitializer AssemblyResolve handler remains as a fallback).
6 $DependencyFolder = Join-Path $PSScriptRoot 'Dependencies'
7 if ($PSEdition -eq 'Core') {
8     $CoreDependencyFolder = Join-Path $DependencyFolder 'Core'
9     $LoaderPath = Join-Path $CoreDependencyFolder 'Microsoft.Graph.Authentication.Loader.dll'
10    if (Test-Path -LiteralPath $LoaderPath) {
11        $null = [System.Reflection.Assembly]::LoadFrom($LoaderPath)
12        [Microsoft.Graph.PowerShell.Authentication.Loader.GraphLoadContextInitializer]::Initialize($DependencyFolder, $CoreDependencyFolder)
13    }
14 }
```

Importing module

```
33
34     # Load the module dll
... 35     $ModulePath = (Join-Path $PSScriptRoot 'Microsoft.Graph.Authentication.dll')
36     $null = Import-Module -Name $ModulePath
37
```

..Authentication.Core is loaded when needed

```
private static Assembly OnDefaultResolving(AssemblyLoadContext defaultContext, AssemblyName assemblyName)
{
    // Only the engine assembly is bridged into the default context. Everything else that Core needs is
    // resolved by GraphAssemblyLoadContext.Load and stays invisible to other modules.
    if (!string.Equals(assemblyName.Name, CoreAssemblyName, StringComparison.OrdinalIgnoreCase))
        return null;

    var context = s_context;
    if (context == null)
        return null;

    string path = context.ResolveManagedPath(assemblyName.Name);
    return path != null ? context.LoadFromAssemblyPath(path) : null;
}
```

Core dependencies

```
protected override Assembly Load(AssemblyName assemblyName)
{
    if (s_sharedAssemblyNames.Contains(assemblyName.Name))
    {
        // Defer to the default context so type identity is preserved across the boundary.
        return null;
    }

    string path = ResolveManagedPath(assemblyName.Name);

    if (s_trustedPlatformAssemblies.TryGetValue(assemblyName.Name, out string runtimePath))
    {
        // This is a shared-framework assembly. Normally the runtime copy must win so that types such as
        // ReadOnlyMemory<T> keep a single identity. However, the module may intentionally ship a newer version
        // (for example System.Text.Json 10.x against PowerShell 7.2/7.4 which only bundle 6.x/8.x). In that case
        // the runtime copy cannot satisfy the reference, so prefer the module's assembly when it is strictly newer.
        if (path == null || !ModuleAssemblyIsNewer(path, runtimePath))
        {
            return null;
        }
    }

    return path != null ? LoadFromAssemblyPath(path) : null;
}
```

Excluded assemblies

```
/// <summary>
/// Assemblies that must always be resolved from the default load context because their types are shared
/// with the cmdlet assembly, the PowerShell engine or the generated service modules.
/// </summary>
private static readonly HashSet<string> s_sharedAssemblyNames = new HashSet<string>(StringComparer.OrdinalIgnoreCase)
{
    "Newtonsoft.Json",
    "System.Management.Automation",
    "Microsoft.PowerShell.Commands.Utility",
    "Microsoft.PowerShell.Commands.Management",
    "Microsoft.PowerShell.Security",
    "Microsoft.PowerShell.ConsoleHost",
    "Microsoft.Graph.Authentication",
    "Microsoft.Graph.Authentication.Loader",
};
```

Q&A

Raise hand
and come to the Q&A area



Stephan van Rooij



Blog & slides



svrooij.io