Git Credential Manager for Windows

The Git Credential Manager for Windows(GCM) provides secure Git credential storage for Windows. GCM provides multi-factor authentication support for Azure DevOps, Team Foundation Server, GitHub, and BitBucket.

Usage

After installation, Git will use the Git Credential Manager for Windows and you will only need to interact with any authentication dialogs asking for credentials. The GCM stays invisible as much as possible, so ideally you'll forget that you're depending on GCM at all.

Assuming the GCM has been installed, using your favorite Windows console (Command Prompt, PowerShell, ConEmu, etc.), use the following command to interact directly with the GCM.

```
git credential-manager [<command> [<args>]]
```

Commands

delete (deprecated)

Removes stored credentials for a given URL. Any future attempts to authenticate with the remote will require authentication steps to be completed again.

This method is being deprecated and users should use "git credential reject" instead

deploy *[–path <installation_path>] [–passive] [–force]*

Deploys the Git Credential Manager for Windows package and sets Git configuration to use the helper.

deploy –path <installation_path>

Specifies a path (<installation_path>) for the installer to deploy to. If a path is provided, the installer will not seek additional

When combined with `--passive` all output is eliminated; only the return code can be used to validate success.

version

Displays the current version.

clear

Synonym for **delete**.

install

Synonym for **deploy**.

uninstall

Synonym for **remove**.

get / store / erase / fill / approve / reject

Commands for interaction with Git.

Configuration Options

Git Credential Manager and Git Askpass work out of the box for most users. Configuration options are available to customize or tweak behavior(s).

The Git Credential Manager for Windows [GCM] can be configured using Git's configuration files, and follows all of the same rules Git does when consuming the files. Global configuration settings override system configuration settings, and local configuration settings override global settings; and because the configuration details exist within Git's configuration files you can use Git's `git config` utility to set, unset, and alter the setting values.

The GCM honors several levels of settings, in addition to the standard local > global > system tiering Git uses. Since the GCM is HTTPS based, it'll also honor URL spe✔ ngs. Reg

`true` `false` `true`

See GCM_MODAL_PROMPT

namespace

Sets the namespace for stored credentials.

By default the GCM uses the 'git' namespace for all stored credentials, setting this configuration value allows for control of the namespace used globally, or per host.

Supports any ASCII, alpha-numeric only value. Defaults to `git`.

letion of credentials even when they are reported as invalid by Git. Can lead to lockout situations once credentials expire and until those credentials are manually removed.

Supports `true` or `false`. Defaults to `false`.

```
git config --global credential.visualstudio.com.preserve true
```

See GCM_PRESERVE

httpTimeout

Sets the maximum time, in milliseconds, for a network request to wait before timing out. This allows changing the default for slow connections.

Supports an integer value. Defaults to 90,000 milliseconds.

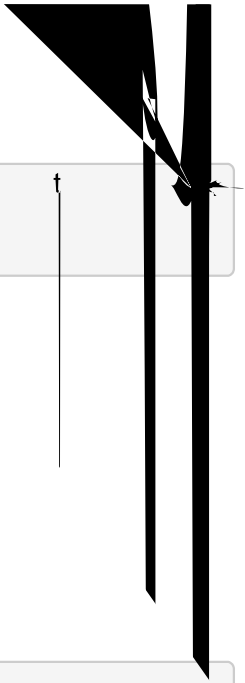
```
git config --global credential.visualstudio.com.httpTimeout 10
0000
```

See GCM_HTTP_TIMEOUT

tokenDuration

Sets a duration, in hours, limit for the validity of Personal Access Tokens requested from Azure DevOps.

If the value is greater than the maximu



```
= true
```

Posh-Git

https://www.nuget.org/packages/Microsoft.Alm.Authentication

The Git Credential Manager does not work on Windows XP, Max OS, or Linux because we had to scope our work and we decided to support the same operating systems that Visual Studio support. Why Visual Studio? Well, because it is our favorite IDE and in order to support Azure DevOps we had to use the Azure Directory Authentication Libraries which only have multi-factor interactive logon support in their .NET libraries. Using .NET means using Visual Studio (which we love anyways) and using Visual Studio means Windows 7 or newer.

Q: Will there ever be support for Windows XP, Mac OS, or Linux?

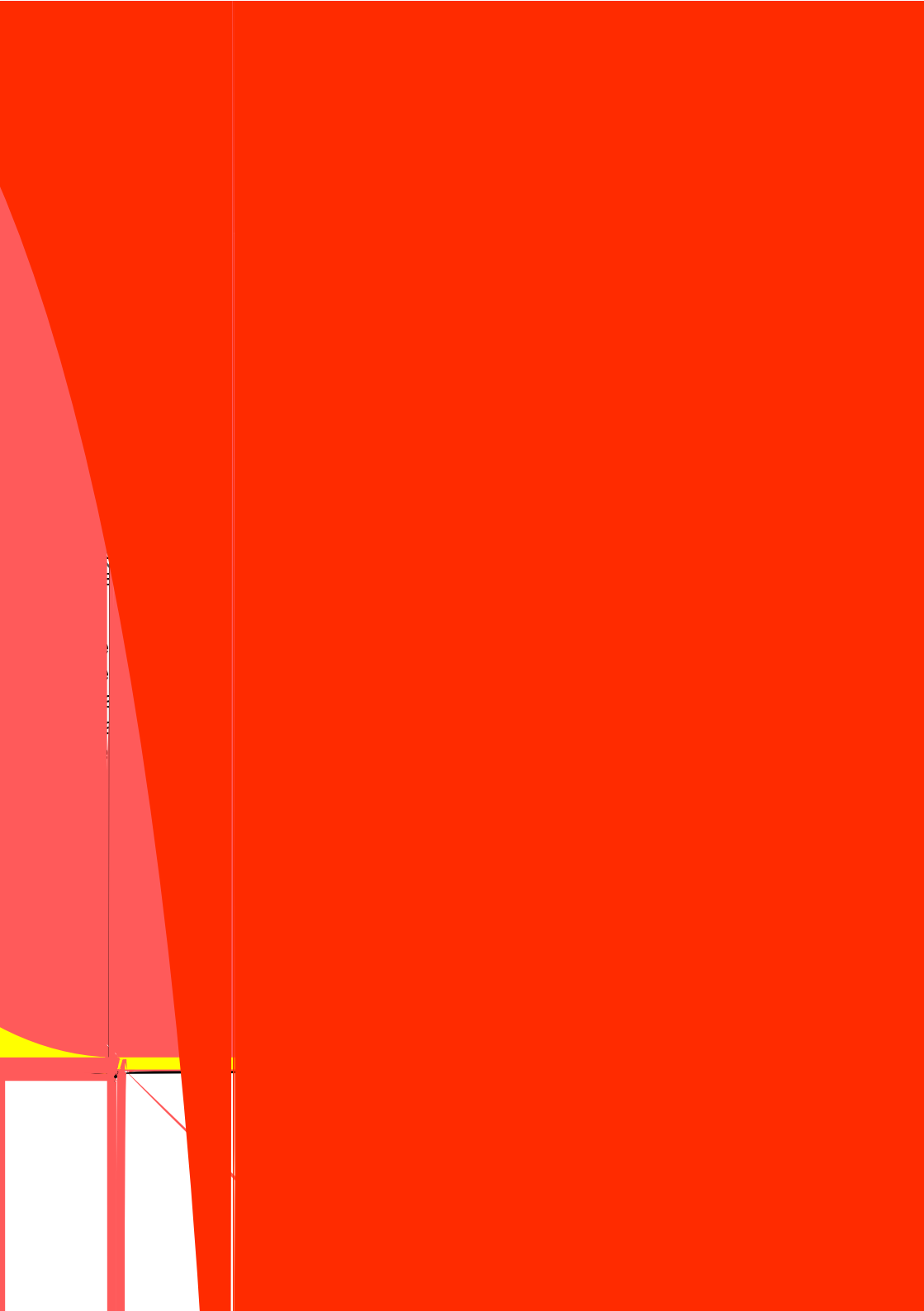
We can safely say that we have no interest in supporting Windows XP. Even Microsoft has ended support for Windows XP. Support for Mac OS and Linux are handled by Microsoft Git Credential Manager for Mac and Linux.

Q: Why is my distribution/version of Git not supported?

nt helpers (`install.cmd` and `GCMW-{version}.exe`) are focused on support for Git for Windows because Git for Windows conforms to the expected/normal behavior of software on Windows. It is easy to detect, has predictable installation location, etc. This makes supporting it easier and more reliable.

That said, so long as your favorite version of Git supports Git's git-credential flow, it is supported by the Git Credential Manager for Windows. Setup will have to be manual, and if you find a way to script it we would love to have you contribute that to our project.

1. Copy the contents of the `gcm-<version>.zip` to your Git's /bin folder. This varies per distribution, but it is likely next to other git tools like `git-status.exe` .
2. Update your Git configuration by running `git config --global credential.helper manager` .



ceTree use GCM?

version of Git installed for the ocal copy of portable Git.

k the ‘Use System Git’

g the GCM in es (but

stead of the SSH URL for `t remote show origin`. The `ttps://` or `http://`. If this is web interface of Azure un `git remote set-url` URL.

ng to ing/unlinking ccount from y?

ccount changes like when rectory (Azure AD), the g a GCM ver

```
Error: cannot spawn askpass: No such file or directory
Error encountered while pushing to the remote repository: Git
failed with a fatal error.
could not read Username for ‘https://******.********.***’: ter
```

token to access

the REST API

```
git config --global credential.validate false
```

Development and Debugging

Developing for GCM and/or Askpass requires Visual Studio 2017 or newer, any version (including the free Community Edition).

Getting Started

The easiest way to get started is to:

1. Install Visual Studio.
2. Clone the repository.
3. Open the solution file (GitCredentialManager.sln) usin

`Program.LoadOperationArguments`

`Program.QueryCredentials`
`OperationArguments`

`GCMW-{version}.exe`

Inno Setup
IDP plugin for Inno
Compiler 5.5.9
Setup

markdown

Pandoc

`.git/`

`SET GCM_TRACE=1`

`SET GIT_TRACE=1`

`setx GCM_TRACE %UserProfile%\git.log`