Terminal v1.0 Roadmap

Overview

This document outlines our roadmap to delivering Windows Terminal v1.0 by spring 2020.

Milestones

The Windows Terminal project is engineered and delivered as a set of 4-week milestones:

Duration	Activity	Releases
2 weeks	Dev Work • Fixes / Features for future Windows Releases • Fixes / Features for Windows Terminal	Release to Internal Selfhosters at end of week 2
1 week	Quality & Stability • Bug Fixes • Perf & Stability • UI Polish • Tests • etc.	Push to Microsoft Store at end of week 3
1 week	Release • Available from Microsoft Store & GitHub Releases (Tues of 4th week) • Release Notes & Announcement Blog published • Engineering System Maintenance • Community Engagement • Docs • Future Milestone Planning	Release available from Microsoft Store & GitHub Releases

Terminal Roadmap / Timeline

Ultimately, we're aiming for Terminal v1.0 to be feature-complete by Dec 2019, and to declare v1.0 by April 2020:

> ⚠ *Note: Terminal v1.0 will be a quality-oriented release driven in large part by the community. So, **if you see bugs, find/file them**!*

Milestone end date	Milestone Name	Key Deliverables
2019-05-07	Announcement	Terminal announced & open-sourced (Build 2019 Terminal session, "Sizzle" video)
2019-07-09	v0.2 (update)	First version of the Terminal released via the Microsoft Store, fundamental features in place, basic tab control, basic UI layout, config & settings via JSON file
2019-08-02	v0.3	Major UI improvements, improved tab bar layout & color, basic a11y

		support, Azure Cloud Shell connection
2019-08-27	v0.4	HTML Copy, Tab Titles, Double/Triple Click Selection, Local Settings, JSON settings validation, A11y improvements
2019-09-24	1909	Stability & Quality improvements, installs Cascadia Code font, adds JSON schema to `profiles.json` settings file enabling Intellisense in VSCode, etc.
2019-10-22	1910	Cascading Settings, Dynamic Profiles
2019-11-19	1911	Final v1.0 feature work
2019-12-17	1912	"Feature Complete" - All v1.0 Features in-place
Winter Vacation	N/A	None planned
2020-01-28	Beta 1	Pri 0/1/2 Bug fixes & polish
2020-02-25	Beta 2	Pri 0/1 Bug fixes & polish
2020-03-24	RC	Pri 0 bug fixes
2020-05	v1.0	Terminal v1.0 Release

GitHub Milestones

Each milestone above is/will be reflected in our GitHub milestones:

Milestone	Description
Terminal-1909	Work planned for 1909
Terminal-1910	Work planned for 1910
Terminal-1911	Work planned for 1911
Terminal-1912	Work planned for 1912
<Future Milestones>	<Coming soon>
Terminal v1.0	Work planned for v1.0, but not yet assigned to a milestone
Terminal Backlog	Work not yet assigned to a milestone or release

Issue Triage & Prioritization

Incoming issues/asks/etc. are triaged several times a week, labelled appropriately, and assigned to a milestone in priority order:

- P0 (serious crashes, data loss, etc.) issues are scheduled to be dealt with ASAP
- P1/2 issues/features/asks assigned to the current or future milestone, or to the Terminal v1.0 milestone for future assignment, if required to deliver a v1.0 feature
- Issues/features/asks not on our list of v1.0 features is assigned to the Terminal Backlog for subsequent triage, prioritization & scheduling.

v1.0 Scenarios

The following are a list of the key scenarios we're aiming to deliver for Terminal v1.0.

> 👉 *Note: There are many other features that don't fit within v1.0, but will be re-assessed and prioritized for v2.0, the plan for which will be published in early in 2020.*

Release	Priority*	Scenario	Description/Notes
V1	0	Performance & Efficiency	Terminal shall be fast and efficient. Input latency should be eliminated wherever possible. Terminal will be very memory-efficient, and will avoid utilizing unnecessary dependencies to minimize memory consumption and disk footprint
V1	0	Reliability	Every reasonable step should be taken to ensure that Terminal will not crash unexpectedly. Crashing is considered harmful to the user's well-being & state of mind. Crashing issues are prioritized Pri-0 by default
V1	0	Code Reuse	Terminal's core engine will reuse & share componentry from within Windows Console wherever feasible to minimize support & maintenance costs for both
V1	0	Terminal Reuse	Terminal's core will be hostable as a UWP (and perhaps WPF) Control so that apps can host/embed a high quality Terminal. This will satisfy a long-standing ask from many customers and partners for a hostable/embeddable Terminal Control.
V1	0	Rich, modern text renderer	Terminal must be able to render glyphs from East Asian and Middle Asian languages, inc. Chinese, Hebrew, Arabic, etc. Terminal will also be able to render Emoji - an increasingly important feature considering that several programming languages now support Emoji in method and variable names! To render such glyphs, the Terminal needs a DirectWrite-based layout & rendering system which supports font fallback, customizable text layout, GPU accelerated rendering, and many other features not currently supported by the built-in Windows Console
V1	0	Solid Unicode & UTF-8 support	Terminal must be able to store data encoded as Unicode UTF-16/UCS-2 and UTF-8, including surrogate pairs. Note: Terminal v1.0 won't be able to support composing characters or grapheme clusters that are not representable with a single unicode codepoint - this will be addressed in a subsequent release
V1	0	International text rendering	The Terminal will support rendering text for almost every language for which there is a fixed-width font including East Asian languages. Bonus points for RTL languages/scripts.
V1	0	Multiple instances	Users must be able to launch multiple independent instances of the Terminal in order to run tools side-by-side / independently
V1	0	Elevation	Terminal can be launched "elevated" with Admin rights if required so that the user can perform operations that affect machine-wide state

V1	0	Multiple Tabs per instance	Each Terminal instance must support one or more independent tabs. This is the #1 ask from the community!
V1	0	Configurability & Customization	The new Terminal will have a modern, flexible settings mechanism that persists settings to/from a JSON file stored in the user's app data folders, and/or in files synchronized between machines via OneDrive, etc. There will be no settings UI in Terminal v1 - this is a feature for a future Terminal release.
V1	0	Accessibility (A11y)	The Terminal will be highly accessible and inclusive. It will expose its contents via UIA to support tools such as Windows Narrator, and UI automation tools including WinAppDriver
V1	1	Color Theming & Styling	The Terminal will honor the user's Windows dark/light theme settings, and/or color accent settings. Also, the Terminal background & text colors will be highly configurable, and importable/exportable via settings files.
V1	1	Background transparency	Background transparency is a valuable feature for many command-line users. Terminal will (optionally) support transparent backgrounds, but without making the Terminal's text content itself transparent (like the Windows Console currently does due to GDI limitations)
V1	1	Fluent "Acrylic" blurred backgrounds	While full transparency is valuable to some, clear/full-transparency can be distracting. Some would like blurred transparency similar to Fluent Acrylic
V1	1	Customizable Key Bindings	Terminal will provide a way for users to customize key bindings, enabling them to configure specific key chords to particular Terminal actions
V1	1	Mouse Support	Terminal will support mouse input, passing mouse movements and actions to command-line apps
V1	2	Azure Cloud Shell	Enable users to register their Azure account/subscription, and allow the Terminal to enumerate and automatically configure a connection to the user's Cloud Shell
V1	2	Multiple panes	Multiple tabs are useful to some, but developers often need to see several files/logs on the same screen at the same time. Windows Terminal should allow a "page" to be split into "panes", each running independent commands/shells/etc. similar to tmux on *NIX/macOS

Feature Notes:

* Feature Priorities:

0. Mandatory
1. Optimal
2. Optional / Stretch-goal