ArcOS
v6

ArcOS v6 Concept

IZAAK KUIPERS

Table of Contents

Foreword

The concepts explained in this document are highly experimental and may or may not make it to production of ArcOS v6. At the time of writing ArcOS v6 development has not officially started nor has any code been written in the name of v6. The concepts, ideas and theories in this document are because of this only a base to concept the general inner workings of ArcOS v6.

Note:	Any questions or concerns can be coordinated with the ArcOS Board exclusively.

From here on out 'ArcOS v6' will be referred to as ***v6***, 'ArcOS API' as ***the API*** and 'Izaak Kuipers' as ***the Author***.

Disclaimer:	None of the information in this document has been coordinated with their respective teams. Such coordination will happen later in this pre-development cycle.

Confidentiality

All information in this document is considered strictly confidential and **cannot be shared with external parties**. ArcOS v6 is the next phase of ArcOS and is therefore still internal until the first beta releases roll out to the public.

Do not mention v6 to external parties, not even the v6 name itself. Screenshots of internal documents, information, discussions or code can result in a termination of your position in ArcOS.

This is not a joke. The existence of ArcOS v6 is considered ArcOS Internal. The public can't know about a fraction of the possibility that v6 is a thing in the future. Keep the knowledge of v6 to yourself and do not share any screenshots related to it.

What does v6 entail?

During the development cycles of ArcOS and its predecessor WebOS, I have performed several rewrites, most of which did not make it to their final stages or anywhere near production ready states. Of all these rewrites, five of them are relevant to the production line of ArcOS. These go as follows:

1. WebOSv1 – June 2020: Vanilla JS, no window, every app was a separate HTML page.
2. WebOSv2 – December 2020: Vanilla JS, windowing system, Apache dependent.
3. WebOSv3 – February 2021: Electron based, Node dependent, more professional.
4. ArcOSv1 – July 2021: rebranding of WebOSv3, major internal improvements.
5. **ArcOS-Svelte-New**: July 2022: complete rewrite, migration to Svelte, backend.

Of these five versions, **ArcOS-Svelte-New** *(now ArcOS-Frontend)* is the current production codebase. All the previous versions have reached EOL one way or another. It is important to note here that the 5.x.x major in the ArcOS versioning system is the way that it is because the current codebase is the fifth notable codebase.

You can already deduce what v6 entails. It will be a complete reimagination of ArcOS as a whole. This means that it will not just be a Frontend rewrite. It will also include better documentation and a reconsideration of our internal infrastructure to improve our own developer experience and system reliability and user experience.

Additionally, the ArcOS **version**, **mode** and **Git build hash** must be easily accessible on the desktop, potentially by something like a **System Information** button on the about page of the Settings app. A one-liner format for the three could be: “ArcOS 6.0.0-release (a1b2c3d)”. where *6.0.*0 is the **ARCOS_VERSION**, *release* is the **ARCOS_MODE** and *a1b2c3d* is the **ARCOS_BUILD**.

Repository location

As with all other ArcOS repositories, v6 will be in the GitHub organization. It will be a separate repository, not another branch of v5. The current predicted location for v6 will be:

IzK-ArcOS/ArcOS-v6

Note:	Any suggestions for the repository path can be coordinated with the Board directly.

Publication to userbase

The v6 repositories will be private at first. This means that the public will not be made aware of the state of v6, and when they are going to be able to use it. This might seem controversial, but it is the way we are going to do it.

As per the open sourcing of v6, I do intend for this to happen, just not before the first stable release has made it to the public. The v6 repositories will remain private until I declare them as stable.

Prereleases?

Yes. There will be a public beta of v6 people can download to check out what is headed their way and to report on any bugs or issues they find. I expect this public beta to happen somewhere after the application system has been finalized, which will be long before v6 is fully refined and ready for release.

License Information

The licenses for ArcOS will remain the same as v5. GPLv3 applies to ArcOS and all its belonging assets. Any external assets (such as auditory and graphical files) will have to be explicitly granted access to before they can be made part of the ArcOS Project. A public policy for this will eventually become available, which contributors and the team will use to acquire licenses for assets before sending them in for addition.

User Support and Documentation

Documentation

One major thing we will be focusing on with the development of v6 is the improved documentation. In early October 2023 I launched the DWoct23F1 practice which allows us to write documentation for v5 in a more manageable way, though it is apparent that we will need more than that to write effective and easy to read documentation for not only the user, but also the developer. With the creation of v6 every developer will keep track of- and write documentation for code they write. A couple rules apply:

- Code documentation **cannot** happen in comments. They only make the code less easy to read. Only use comments if you need to make something clear that would otherwise make no sense to others.
- These snippets **do not have to be long**. 50 words is already enough. Just explain **what the function or component is**, **what it does**, and **how it can be implemented**.
- Write these snippets in a **logical and consistent sense**. Write every snippet in **the same format** to make it clearer for DWC when reading through multiple at once.
- Coordinate with DWC about new snippets and share said snippets with them immediately along with where exactly the code is in the codebase so that it can be added to the *wiki* in the GitHub organization.
- Before documenting existing code, check if somebody else did not already do so. Look through the documentation and find snippets of the code before documenting it yourself. If nobody has written the snippet in question, you may write it yourself or contact me about it so I can assign somebody from DWC.

Bug Reports

In v5 we introduced BugRep which is our integrated system for dealing with bugs and solving them in the most efficient way. All crashes, bugs and miscellaneous problems are sent to the BugRep servers for us to solve. This **will not** change in v6. The same Pocketbase server will be used to host BugRep, and we will be using the Pocketbase SDK and the same code from v5 to create, get, check bug reports, detect crashes, and to collect system information to put in the report. This means that the following data sources need to still be present in v6:

- **UserData**: the global writable containing the data of the user.
- **UserName**: the global writable containing the username.
- **LogStore**: the logs the frontend generates.
- **Servers**: the stores for the different servers a user can connect to (ConnectedServer comes to mind).

End Of Life

Starting with v6, versions will be able to go End of Life if a new version is released to replace it. DT can reference design **v6_EOL** which illustrates what an EOL notification will look like. This EOL system will either be handled through Pocketbase or through a custom API.

Method: Pocketbase

A new collection would be created on the ArcOS Pocketbase Instance. Its name would be something along the lines of “eol” and its API rules would specify that everything can *read* the EOL records.

Each EOL record would contain a couple properties:

- The version number of the version that will go EOL.
- The date from which the EOL is effective.
- Its ID will still be randomly generated by Pocketbase, as is the case with all other records on ArcPB.

Notification

Starting with ArcOS 5.0.12, the frontend will check for an EOL record that matches the version number of the frontend when the desktop or ArcTerm Mode. If such a record exists, check if the date on the record is in the past, and display a dialog box informing the user that it is recommended to update.

Limiting EOL versions of ArcOS?

As of right now, ArcOS versions that are out of support will not receive any kind of graphical limitations like the Firefox detection does. BugRep will however require a “revision” to be specified alongside the data of any request, which will increment with each new version. This means that if a version of ArcOS goes EOL, we will no longer receive Bug Reports from those versions. The Bug Reports App in ArcOS would ask them to update ArcOS to re-enable bug reporting.

Gradual dysfunction

ArcOS EOL is there for a reason. As ArcOS progresses, the backend will slowly but surely change, causing these older versions to lose more and more features as time goes on. The End-of-Life system has been put in place so that users will more quickly and effectively migrate to newer versions of ArcOS, preventing them from reporting issues of already solved bugs, and experiencing unexpected behavior none of us can account for.

Browser Information

Browsers discontinuation?

ArcOS v6 will not feature the discontinuation of browser support. ArcOS Legacy was an Electron-only application, but I changed that in v5 with the introduction of Svelte. Tauri integration came next, which was later once again replaced with Electron. ArcOS v6 will feature both Desktop and Browser apps, with the desktop app having a different ID to differentiate it from the legacy-electron, v5-electron and v5-tauri apps. This is to allow users to run the older versions of ArcOS side-by-side. This has no practical use case besides providing a way to visualize the evolution of the project.

Browser Support

With v6 I aim to support as many browsers as we can. Firefox support may not return due to its lacking CSS features and general compatibility, but Safari needs to be supported for the Apple users of our userbase. For this we will need a MacOS KVM to reliably and effectively test the ArcOS Frontend frequently to ensure it works on stock Safari.

What about other browsers?

Browser testing of Opera, Vivaldi, Brave, etcetera does not have to happen individually because they are all Chromium-based and therefore can be evaluated by testing Chrome and/or Chromium instead.

Legacy Browsers

Because we will be using Svelte, legacy browser support such as Internet Explorer and older versions of Firefox (pre-68) will not be officially supported by v6, which is exactly as it is right now. Implementing support for legacy browsers will only cover a fraction of the userbase and we need to save up development time to use on other aspects of v6.

Note:	You can expect no support for outdated browsers to come to v6.

Mobile?

No. My initial vision for ArcOS still stands. It is a *desktop* operating system and is therefore not designed for – or supported by – mobile devices. It is meant to run on Desktop Computers, Laptops or Tablets exclusively. I do not intend nor support any efforts towards mobile support from anyone on the development team, I will not even permit it even if it is considered "a personal project."

Browser Testing

A list of browsers will be formulated that will require individual initial testing right before 6.0.0, and any following versions, are released. After that all Chromium-based browsers will be tested at once by testing Chrome and/or Chromium.

A list of browsers that will be tested follows:

- Chrome/Chromium — Chromium
- Opera (GX?) — Chromium
- Vivaldi — Chromium
- (The New) Microsoft Edge — Chromium
- Brave — Chromium
- Safari — **WebKit**

Mobile browsers **will not be tested** even though tablet devices are technically supported by ArcOS. With that said, desktop browser testing will already cover 99% of cases related to mobile devices (tablets *exclusively*).

Browser behind Desktop App

As said in other parts of the document, Electron is and will remain to be the runtime behind the ArcOS desktop app. This means that the ArcOS-Electron-Compiler repository will simply be updated to house v6 instead of v5, perhaps on separate branches.

Designing

Design Migration

As previously discussed in Developer Chat 2, we will be redesigning a great portion of ArcOS before adding their respective features to v6. During the migration process, all visual components of v5 will be examined to determine whether it's sufficient to add to v6. Anything that does not meet the criteria for addition will either be excluded from v6 entirely or have their designs redone to meet the criteria either way.

Design Principles

The design principles will be created by me and DT once the development process starts. This design principle will focus on a professional but also playful visual environment for ArcOS and consistency between the different states (think about server select, login and desktop, they must be fully consistent even though the login currently differs).

The glass-look and pronounced borders will remain. Animations can be adjusted to be more impressive though we do have to remember not to make it *too childish*. ArcOS is turning from a hobby project to something way more professional, and this needs to adhere to that notion as well.

Logo Redesigns

ArcOS v6 won't feature redesigned logos for the different modes. The current logos (at least those that correspond with modes that won't be going away) will remain the same in ArcOS v6. I don't see the necessity for a redesign of the primary or secondary branding.

Branding

At the current time there is no need to change the branding of ArcOS, though it might be preferable to give the different modes of ArcOS (release, development, etcetera) separate badges in their icons alongside the different coloring. I will coordinate with DT to get these in ArcOS Conceptual.

A change in ArcOS Modes will occur. Several modes will be removed or changed. These changes will be implemented with the release of ArcOS 6.0.0:

- **ADMIN** Removed.
- **SIEGE** Removed.
- **UNSTABLE** Changed: needs a better representation of an unstable state.
- **UWU** Removed.

Frontend Structure

Git Submodules

Starting with v6, ArcOS will be making use of Git Submodules in the frontend to further separate the different aspects of the codebase. Each state will be its own repository and its own submodule inside of the main frontend. Each application will also be its own submodule.

This will require us to use the recursive flag when pulling ArcOS to clone all the submodules along with the primary module itself. Otherwise, the frontend will fail to find the states and it will crash instantly, which it will not be able to do, because the *crash state* will also be missing.

States

The state management system of v5 will be rewritten from the ground up. A change was made regarding this system that eliminated the need for Svelte Writables for changing states. For the uninitiated, states are the different phases ArcOS can display. This includes the following list of states:

- Boot — The boot screen.
- FTS — The First Time Setup
- Server Select — the Multi-API Server Selector
- Login — The Login Screen
- Desktop — The ArcOS Desktop itself
- Turned Off — The state that closes ArcOS or shows an image instead.

The states **Shutdown**, **Restart** and **Logoff** all use the same **Login** component, with different parameters passed to it to tell the login what it's supposed to display and perform.

The Login

The ArcOS login screen will not change at all in v6. This is because a rewrite already happened recently that implements the stability, we are already hoping to achieve with v6. A rewrite of the login at this point would simply be a waste of time.

Reference:	For more information: The Login rewrite happened here.

What might have to change though are the functions used for dealing with API interfacing and user authentication since I do imagine these to be completely redone in v6.

Backend

ArcAPI v2

ArcOS v6 will not come with a new API. What we do have to do is document the backend code as best as we can. This must be finished before the release of ArcOS 6.0.0. To accomplish this, I expect DWC to work together with BDT to write professional documentation that is easy to understand for the other members of the ArcOS Team and any external parties.

API Interfacing

In v6 the way the frontend interfaces with the API will be changed. There should be no direct mentions of endpoint paths in code throughout ArcOS. All API interfacing will be channeled through the ArcAPI SDK.

This does mean that the SDK will have to be completed before the v6 frontend can begin, since the SDK will be a vital part of the frontend.

Registered APIs

Starting with ArcOS v6, users will be able to select from a set of publicly available APIs for different people. The APIs available will still have to be decided, but confirmed right now is:

Server	*Description*	*Existing?*
community.arcapi.nl	ArcOS Community API	Yes
endermanlair.arcapi.nl	Enderman's Lair API	Yes

ArcOS Applications

Application Sideloading

With v6 we're recreating the base concept of ArcOS. This includes the way we load and handle applications. Application sideloading has been a major conversation starter since the very first members of the community joined, and it is something that I have feared for many reasons, mainly because Svelte does not make it easy. Since we will be redoing the entire frontend, it might be a good option to attempt another implementation of application sideloading. This would allow the community to make and submit applications to ArcOS for other people to download and install onto their existing ArcOS installation.

Svelte Custom Elements

By setting a flag in the configuration files of a Svelte Vite project we can turn any Svelte project, including ArcOS itself, into a custom HTML element that can be loaded in by putting its designated tag in the HTML after the module has been imported. This is a way to create at least the window body. What we must consider here is:

1) How is the sideloaded application going to interact with the ArcOS API, more specifically how do we pass the authentication down to the application?
2) How will the window metadata be defined? Inside the module itself or outside in another JSON file?
3) Images will have to be stored directly in the JS by using data-strings. This threshold can be modified manually in the Vite configuration, I will set the maximum file size for inlining to 10MB.
4) CSS will too be directly included in the JS bundle. This does mean that app developers will have to use the CSS style-tags (preferably *scoped* styles) to style their app components instead of importing external CSS files.

The caveat with this is obviously, file size. The compiled JS bundle can count into the megabytes in size, depending on how many assets are bundled in the program.

This is one of the changes that isn't definitive yet. I'm not sure if application sideloading is viable, but it would allow for more flexibility and development capacity because external parties will be developing a good portion of the applications instead of FDT and DT.

Moderation

We must keep in mind that people will have the opportunity to upload the weirdest and most inappropriate material to ArcOS if Application Sideloading comes with public repositories people can submit to. We will have to devise a team specifically to maintaining and moderating the ArcOS app repositories to prevent people from breaking the rules.

“App Store”

It may finally be a good idea to implement some sort of App Store that people can use to easily browse and install applications. This will have to implement some sort of paging system and a robust API in the background because third-party apps will not be decentralized as opposed to ArcAPI.

App Runtimes and Global Variables

In ArcOS v5 all applications used globally stored variables initially before I rewrote a portion of them to use the App Runtime class. Starting with v6 all applications will be required to use the App Runtime classes for them to make their way into ArcOS. This also ensures that multiple instances of the same application are possible because two instances won’t rely on the same variable definitions.

The App Runtime class itself won’t be that different from its current implementation, except for the fact that it will be a **required** property of an app’s data unless it is a sideloaded application. Sideloaded applications don’t need App Runtimes because their variables are isolated from the rest of the applications anyway because of the way that sideloading will work with the Custom Svelte Elements mentioned in Svelte Custom Elements.

Multiple Application Instances

In early July of 2023, Ivan of BDT attempted a fork of v5 in which he was going to implement the ability to open multiple instances of the same application at once. This introduced the **Unstable** branding because the frontend was not at all stable and it was unviable to keep that development cycle going. That frontend has since been terminated, but its core principle remains. ArcOS v6 will allow for multiple instances of the same application to be opened. This includes built-in applications created by DT and FDT, but also third-party applications created by the ArcOS Community.

Window Data Inheritance (WDI)

Every instance will contain a direct uninherited copy of the original window data along with a Process Identifier (or PID). This allows us to modify each instance of each application separately without having to worry about unwanted object inheritance. Each instance will also have a separate invocation of the Svelte component that is the window content.

ArcTerm

With v5 ArcTerm got a major refactor that still stands strong to this day. It is built consistently in a robust fashion and rarely crashes. It is already using classes and inheritance so we can easily copy ArcTerm from v5 to v6 without any updates or rewrites.

Command Compatibility

The compatibility of commands between v5 and v6 is going to be sparse. Because we are rewriting the entire frontend from scratch, many – if not all – functions, variables and code locations are going to be different between v5 and v6. Because of this we have a couple options to tackle this problem:

Rewrite all commands from scratch.

Keep the existing commandset but rewrite each command from scratch to potentially even improve code quality and crash resistance. This does massively increase development time because of the sheer number of commands in ArcTerm, though it might be the best option from a user experience point of view.

Update existing code to work with new codepoints.

Update all existing commands to work with the new codepoints by using Visual Studio Code and IntelliSense to automatically import a lot of the missing variables and functions, assuming their names and syntaxes remain the same. This can cut down development time, but any bugs that we have not discovered in v5 might make their way to v6.

Start a new commandset.

Start from scratch. Keep ArcTerm itself but rewrite the commandset along with the commands themselves. It would be like how ArcTerm was created back in 2021 after the port of NetCMD: all new commands in an all-new environment.

ArcTerm Mode

ArcTerm Mode has been a way for the user to access ArcOS resources without having to load the desktop. I intend to keep this functionality intact in v6. In v6 you will too be able to press **Alt+F8** on the boot screen to launch ArcTerm mode. I do not expect this or the rest of ArcTerm to change much during the rewrite process.

What will change?

ArcTerm mainly needs to have its file structure addressed. The folder of ArcTerm itself will change from **terminal** to **ArcTerm** (actual folder location still unknown), with its CSS files more organized and better utilized.

We also must account for the API interfacing changes mentioned earlier. The ArcTerm functions will be copied directly from v5 and will have to be modified in order to work with the new interfacing method.

ArcTerm Isolated

With v6 comes the **discontinuation of ArcTerm Isolated**. I have made this decision because it is far too difficult to maintain and only a fraction of the userbase makes use of it effectively. ArcTerm Isolated has less features than ArcTerm Mode of the ArcOS Frontend so I don't see a reason to keep it supported in the long term.

ArcTerm Commands

Below follows a list of commands that will be in v6. Most of these are already present in v5, and this list will likely change around quite a bit during the development process of v6.

Command	*Description*	*Command*	*Description*
arcfetch	System information	goto	Goto a given section
cd	Change directory	end	End command exec
clear	Clear terminal	goose	Goose bumps
dir	List directory	useradd	Create a user
echo	Echo given text	sud	Set User Data
vars	Display variables	sleep	Sleep for Xms
exit	Exit terminal	bgs	List wallpapers
help	Show list of commands		
history	Show command history		
logout	Logout ArcOS		
ls	List directory		
mkdir	Create directory		
rm	Delete file/directory		
reload	Reload Terminal		
restart	Restart ArcOS		
shutdown	Shutdown ArcOS		
indesktop	Check if in desktop app		
ver	Display version		
reset	Reset ArcOS instance		
rf	Read a file		
ri	Display an image		
users	List users on ArcAPI		
read	Read from user input		
set	Set a variable		
soundbus	Access SoundBus class		
config	Show ArcTerm config		
exec	Execute a file		
verbose	Enable verbosity		
desktop	Switch to desktop		
servers	Show servers		
if	If statement		