**CBILITY Technologies Co., Ltd. - Animal Shelter Management**

**Software Deployment Plan**

**2025-10-20**

**1. System Requirements**

Operating Systems:

* Windows 10 or 11
* Linux SuSE Leap 15 and above
* Linux Ubuntu 14 and above
* Other Debian-based and Red Hat-based Linux may work but are not officially tested.

CPU: amd64 (x86\_64) architecture

RAM: at least 2 GB (4GB recommended)

Storage: at least 18MB for the application (excluding user’s content such as animal information)

**2. Deployment Strategy Summary**

The desktop application framework that we use, Tauri, has built-in capability to build and package the application into various formats without additional tools (in most cases\*).

The format depends on the target operating system.

* EXE: for Windows
* RPM: for SuSE Leap (and other Red Hat-based Linux)
* DEB: for Ubuntu (and other Debian-based Linux)

\* To build a Windows installer on Linux, additional tools NSIS and cargo-xwin need to be installed on the development system.

**3. Installation Package Contents**

***3.1 Required source or compiled files***

For Linux (RPM and DEB):

* The compiled executable called “animal-shelter-manager”
* The launcher called “animal-shelter-manager.desktop”

For Windows (EXE Installer):

* The compiled executable called “animal-shelter-manager.exe”
* Additional components used by the installer. These files are automatically generated and will be deleted with the installer. Examples are:
  + System.dll: used by the installer to invoke system calls
  + nsDialogs.dll: used by the installer to build dialog windows
  + modern-wizard.bmp: installer’s UI background

***3.2 Required third-party components***

For Linux:

* WebKitGTK: One of Linux’s most common web-based UI rendering engines. This is shipped with almost all modern desktop linux distributions such as OpenSuSE Leap 15+ and Ubuntu 14+. If it is, for some reason, not installed by default or deleted by the users, they will have to install it separately, but this is very rare.
* GLIBC: Standard wrapper around Linux system calls.

For Windows:

* WebView2: Window’s native web-based UI rendering engine. This is shipped with all Windows 10 and 11 by default but if it has been deleted, the installer will automatically install it.

***3.4 Required graphical assets, configuration and other non-program files***

* App icons of various sizes as recommended by each package format

***3.5 Documentation files to be provided***

* README.md

***3.6 Development files and components that must be excluded***

* All files except for the compiled executable and graphical assets (icons)

**4. Additional Code Required for Deployment**

None

**5. Deployment Tasks**

These steps assume that the build process for all targets are performed on a Linux host machine.

For Linux:

Since GLIBC tends to be backward compatible but not forward compatible, Tauri recommends building the application on the oldest version of the operating systems that we intend to support. Thus, our approach is to use VMs running those OS versions to build the application instead of the host machine.

1. Install KVM if it is not already installed.
2. Download OS image for Open SuSE Leap 15 Desktop and Ubuntu 14 Desktop from the official websites.
3. Using KVM, create a VM running Open SuSE Leap 15 Desktop.
4. On the VM, follow Tauri development prerequisites. This involves installing the dev versions of packages such as openssl, webkitgtk, as well as installing Rust. Git must also be installed if it is not installed by default.
5. Clone the project from Github remote repository on to the VM.
6. Run Tauri build and bundle commands, set the target to “rpm”. The full command is “cargo tauri bundle --bundles rpm”. This process creates a RPM package in the “release” directory.
7. Use SCP command on the host machine to copy the RPM package to the host machine.
8. Close this VM. Then, create another VM running Ubuntu 14 Desktop.
9. Repeat steps 4 and 5 on the new VM.
10. Run the Tauri build and bundle command, set the target to “deb”. This process creates a DEB package in the “release” directory.
11. Use SCP command to copy the DEB package to the host machine.
12. The copied DEB and RPM packages will be tested and published subsequently.

For Windows:

Building a Windows installer on a Linux machine is officially supported by Tauri, and it is guaranteed to run on both Windows 10 and 11. Thus, VMs are not needed.

1. Install NSIS, a tool for creating Windows installers.
2. Add “x86\_64-pc-windows-msvc” as Rust compilation target.
3. Install “cargo-xwin” (Rust’s Windows SDK) with the command “cargo install --locked cargo-xwin”.
4. Run the Tauri build and bundle commands, set the runner to “cargo-xwin” and target to “x86\_64-pc-windows-msvc”. The full command is “cargo tauri build --runner cargo-xwin --target x86\_64-pc-windows-msvc”. This process creates a Windows installer in the “release” directory.
5. Use SCP command to copy the installer to the host machine.
6. This installer will be tested and published subsequently.

**6. Deployment Test Plan**

We will continue to use VMs to test the packages on various OS versions. The key distinction between this VM and the ones used for building applications is that this VM must NOT have any additional packages installed, as it is intended to represent the “default” state of the OS.

For Linux:

1. Create 2 VMs running Open SuSE Leap 15 and 16.
2. Install the package on both VMs.
3. Perform functional tests on the application.
4. Create 7 VMs running Ubuntu 14, 16, 18, 20, 22, 24, and 25.
5. Repeat steps 2 and 3 on these VMs

For Windows:

1. Install the package on 2 actual Windows machines, one running Windows 10, another one running Windows 11. VMs running windows are acceptable alternatives.
2. Open the installers and follow the installation steps.
3. Perform functional tests on the installed applications.