Project Horizon: Requirements

A. Wilcox, Adélie Linux
Tambra Wilcox, Adélie Linux
Elizabeth Myers, The Interlinked Foundation
Jeremy Rand, Namecoin
Lee Starnes
Max Rees
Samuel Holland
Luis Ressel
Alyx Wolcott

Project Horizon: Requirements
by A. Wilcox, Tambra Wilcox, Elizabeth Myers, Jeremy Rand, Lee Starnes, Max Rees, Samuel Holland, Luis Ressel, and Alyx Wolcott

Publication date 2019

Abstract

This document describes the functional and non-functional software requirements for Project Horizon, the installation system for Adélie Linux.

Table of Contents

List of Tables

Chapter 1. Introduction

Purpose

This document describes the functional and non-functional software requirements for release 1.0 of the Project Horizon system. Unless otherwise noted, all requirements specified in this document are of high priority and required for release 1.0.

Intended Audience

This document is intended to be used by developers to implement Project Horizon 1.0. This document is additionally intended to be used by the wider Adélie Linux community to determine how Project Horizon is designed, its intended purposes, and for development of future releases of Project Horizon.

Project Scope

For information about the scope and overall feature set of release 1.0 of Project Horizon, see Project Horizon: Vision, 2019, Wilcox et al.

References

1. *Wilcox et al. (2019), Project Horizon: Vision.* On-line reference: on the World Wide Web at https://www.adelielinux.org/horizon-vision/ [https://www.adelielinux.org/horizon-vision/], or in the Horizon Git repository at `devel/vision/`.

2. *Adélie Linux Platform Group (2019), Platform Group Documentation.* On-line reference: on the World Wide Web at https://wiki.adelielinux.org/wiki/Project:Platform_Group [https://wiki.adelielinux.org/wiki/Project:Platform_Group].

3. *FreeBSD (2011), crypt(3).* On-line reference: on the World Wide Web at https://www.freebsd.org/cgi/man.cgi [https://www.freebsd.org/cgi/man.cgi?query=crypt&sektion=3&manpath=FreeBSD+9.0-RELEASE], or on most Linux or BSD computers by running **man 3 crypt**.

4. *Wilcox (2019), Programming Languages supported by Adélie Linux.* On-line reference: on the World Wide Web at https://code.foxkit.us/adelie/packages/wikis/Programming-Languages [https://code.foxkit.us/adelie/packages/wikis/Programming-Languages].

5. *Wilcox et al. (2019), HorizonScript Specification.* On-line reference: on the World Wide Web at https://www.adelielinux.org/horizon-script/ [https://www.adelielinux.org/horizon-script/], or in the Horizon Git repository at `devel/script/`.

Chapter 2. Overall Description

This chapter provides a high-level description of Project Horizon. For more information on the background of Project Horizon, see Project Horizon: Vision, 2019, Wilcox et al.

Project Perspective

Project Horizon is a new system that provides computer users and administrators the ability to install the Adélie Linux operating system. It is intended to replace the current manual installation procedure for the use cases identified in *Project Horizon: Vision, 2019, Wilcox et al sections 1.2.2 and 1.4*.

It is expected over the next few years that Project Horizon will evolve with the Adélie family of software and services to provide a well-accepted libre environment for general purpose computing.

User Classes and Characteristics

The identified user classes are defined in *Project Horizon: Vision, 2019, Wilcox et al section 1.4*. A summary of these classes is included below.

Table 2.1. User classes for release 1.0 of Project Horizon

User Class	Description
The Beginner "Charlie" (favoured)	This class of users has never used Linux before, but has used another operating system on their computer. Users of this class will need documentation that contains clear descriptions of what their option selection will cause to happen. Users of this class typically will want to keep their existing operating system while installing Adélie Linux to a second partition or disk.
The Enthusiast "Dakota" (favoured)	This class of users has experience with other Linux distributions, and wishes to install Adélie Linux. Users of this class will want to feel in control of their computer and the system being installed to it. Users of this class will use the majority of the features denoted as "advanced" in this requirements specification.
The Administrator "Jamie"	This class of users is interested in using HorizonScript to automate installation of Adélie Linux to one or more computers. Users of this class may or may not use the Horizon UI to create the HorizonScript.
The Developer "River"	This class of users has significant Linux experience and wishes to install Adélie Linux on more "esoteric" hardware, such as a BeagleBone Black or Pentium III.

Operating Environment

OE-Defined. In this document, the term "installation environment" refers to the Horizon UI or Horizon Runner executing on the target computer that will install Adélie Linux. The term "runtime environment" refers to the Horizon UI or Horizon Runner executing on a computer that is already running an existing OS.

OE-1. Project Horizon shall run on any computer that is supported by the Easy Kernel and is using a Tier 1 supported CPU architecture as defined at Platform Group Documentation, 2019, Adélie Linux.

OE-2. Horizon UI shall run under the X11 display server with a screen resolution no smaller than 800x600.

OE-3. The system shall provide the adelie-base-posix package in the installation environment.

Implementation Constraints

IC-1. Project Horizon code written in C-based languages shall conform to the Adélie Linux code style .

IC-2. Horizon UI shall be written in C++ using the Qt widget toolkit.

IC-3. The system shall use the Modular Crypt Format as described in crypt(3), 2011, FreeBSD and as implemented in the musl C library to encrypt passwords with the SHA-512 and/or Blowfish algorithms.

IC-4. Horizon Runner shall be written in a programming language that is present in the System repository of Adélie Linux, as listed at Programming Languages, 2019, Wilcox.

IC-5. The system shall perform all communication with external packages via documented APIs, and shall not use pipes to user-facing commands.

Assumptions

ASM-1. The Perform Installation and Installation Completion sections of Horizon UI are assumed to be their own executable, which is run when either: the Horizon UI completes gathering information from the User and has created the HorizonScript to use for installation, or an Administrator has started Horizon automatically and wishes to have graphical output of the installation progress.

Project Dependencies

DEP-1. The Horizon UI depends on the Horizon Runner.

Chapter 3. Functional System Requirements

Horizon UI

The Horizon UI is the wizard-style user interface component of Project Horizon. The system takes the input of the User and creates a HorizonScript. The resultant HorizonScript is either executed on the running system when started in an installation environment, or saved for later use when started in a runtime environment.

System Initialisation

Description / Priority

This section describes the boot-up and initialisation procedures for the Horizon UI Installation Environment.

Functional Requirements

UI.Early.LoadFirmware. If the User has specified to load firmware in the installation environment, the system shall install the linux-firmware package to the installation environment before eudev is started.

UI.Early.DetectGfx. The system shall detect all usable graphics adaptors present in the computer.

UI.Early.NoGfx. If there are no usable graphics adaptors present, the system shall display a message that installation cannot continue, and allow the User to login to a virtual TTY or reboot the computer.

UI.Early.MultiGfx. If multiple graphics adaptors are present, the system shall attempt to determine which graphics adaptor has a monitor attached. If the determiniation is that multiple graphics adaptors have monitors attached, or it is indeterminiate which adaptor has a monitor attached, then the system shall prompt the User to choose which graphics adaptor to use for installation.

Global UI Requirements

Description / Priority

The system must obey these requirements on all screens, except where explicitly noted.

Functional Requirements

UI.Global.Cancel. The system shall display a Cancel button on every screen.

UI.Global.Cancel.Confirm. If the User chooses the Cancel button, the system shall confirm that the User wishes to cancel installation. The system shall additionally explain that no data has been modified yet, and that cancelling installation will cause the computer to reboot.

UI.Global.Cancel.Perform. If the User chooses to confirm cancellation of the installation, the system shall reboot the computer.

UI.Global.Back. The system shall display a Back button on every screen except the initial screen, which will move the User back to the previous screen.

UI.Global.Back.Save. If the User chooses to go back to a previous screen, the system shall save any changes the User has made to the current screen before moving back to the previous screen. The system shall then redisplay those changes when the current screen is shown again.

User Language Selection

Description / Priority

A User must determine what natural language they are most comfortable with using for installing Adélie Linux. The system will allow the User to select their preferred natural language from a list of the languages to which the system has been translated. Priority: Medium.

Prerequisites

UI.Language.Prerequisites. The system must have more than one language translation available for use.

Functional Requirements

UI.Language.List. The system shall present the User with a list of natural languages, in the form of: "Install Adélie using English", "Installer Adélie en français".

UI.Language.Button. For this screen only, the system shall present the "Next" button as a single right-facing arrow. The system shall not present the Cancel button on this screen.

UI.Language.Enable. When the User chooses a language from the list, the system shall enable the Next button to be clicked.

HorizonScript Keys

Script.Key.language. `language` — The locale identifier of the chosen language.

Initial Introduction

Description / Priority

The system will explain to the User an overview of the installation process, and introduce common UI elements. A User may choose to run a tool or proceed with the installation.

Functional Requirements

UI.Intro.Describe. The system shall present the User with a greeting message including a high-level overview of the installation process. The system shall reassure the User that the installation process will not modify any data until the final step.

UI.Intro.Elements. The system shall describe the UI elements common to each screen, including the Help button, the Back/Next buttons, and the Cancel button.

UI.Intro.Tools. The system shall allow the User to start a tool from the following list before beginning the installation.

1. A terminal.
2. An external partition editor.
3. A lightweight Web browser.

Input Device Configuration

Description / Priority

A User may have a keyboard that is not the Linux default of US QWERTY. As such, a User must be able to choose a keyboard layout that matches their hardware.

Functional Requirements

UI.Input.LayoutList. The system shall present the User with a list of available keyboard layouts.

UI.Input.ChooseLayout. When the User selects a keyboard layout, the system shall load the keyboard layout and apply it.

UI.Input.Test. The system shall present an input box for the User to test their selected keyboard layout.

HorizonScript Keys

Script.Key.keymap. `keymap` — The identifier of the chosen keyboard layout.

Disk Partitioning Setup

Description / Priority

The User must partition their hard disk drive to be able to install Adélie Linux to it. The system will allow the User to do so, using automatic partitioning based on the platform and disk size, or allowing the User full control over partitioning. (Priority: High)

The system will additionally allow the User to set up a dual-boot system, if the disk has enough free space to create the requisite partitions for an Adélie Linux installation. (Priority: Medium)

The system will *not* allow the user to resize or delete existing partitions.

If the system is running in a runtime environment, the User must describe the disks and any operations to perform on them.

Functional Requirements

UI.Partition. The system shall allow the User to partition their block devices.

UI.Partition.Runtime. If the system is running in a runtime environment, the system shall prompt the User to enter details about the block devices present on the target computer.

UI.Partition.Runtime.DiskDetails. The system shall prompt the User to enter the path to the block device, the size of the block device, and an optional identification string which can be used to ensure the proper block device is selected on the target computer.

UI.Partition.Runtime.Labels. The system shall allow the User to choose whether to retain an existing disklabel on the block device, or create a new one.

UI.Partition.Runtime.LabelType. The type of the disklabel on the block device must be specified, and of type APM, MBR, or GPT.

UI.Partition.Runtime.Partition. The system shall allow the User to create partitions on the specified block devices.

UI.Partition.Runtime.Partition.ExistingLabel. If the User has specified to retain an existing disklabel on a block device, the system shall stress to the user that they may only create new partitions on the disk and must not specify existing partitions on the disk.

UI.Partition.Runtime.FS. The system shall allow the User to choose to create new file systems on any named block device, including partitions that may not have been specified (since they may already exist on the block device).

UI.Partition.Runtime.Mount. The system shall allow the User to choose what block devices to mount, and the mount point to use for each mounted block device.

UI.Partition.Runtime.Mount.Options. The system shall allow the User to specify custom mount options for each mounted block device.

UI.Partition.Install. If the system is running in an installation environment, the system shall gather details about available block devices on the computer.

UI.Partition.Install.Ignore. When gathering details about available block devices on the computer, the system shall ignore optical disc and tape drive units.

UI.Partition.Install.Details. The system shall gather the following details about available block devices: identification string (manufacturer, model), total size, current disklabel or file system (if any).

UI.Partition.Install.ExistingLabels. If the system contains one or more block devices with disklabels, the system shall gather the partition layout of each disklabel, and file system and type code information for each partition.

UI.Partition.Install.ExistingOS. The system shall use the `os-prober` tool to determine if any operating systems are currently installed on the computer.

UI.Partition.Install.FreeSpace. The system shall determine if any block device with existing disklabels has enough free space in the existing disklabel to create the requisite partitions needed for an installation of Adélie Linux.

UI.Partition.Install.UserPrompt. The system shall prompt the user for the desired partitioning action to take.

UI.Partition.Install.UserPrompt.MultiDisk. If the computer has multiple available block devices, the system shall allow the User to choose which block device to use for installation, and display the appropriate screen based on the block device chosen.

UI.Partition.Install.UserPrompt.SingleDisk. If the computer has only one available block device, the system shall display the appropriate screen based on that block device.

UI.Partition.Install.UserPrompt.NoDisk. If the computer has no available block device, the system shall inform the User that no block devices are available, and that installation cannot continue.

UI.Partition.Install.UserPrompt.Unlabelled. If the block device contains no label, the system shall allow the User to choose an automatic disk label and partition layout based on the platform or to specify their own partition layout.

UI.Partition.Install.UserPrompt.LabelledWithSpace. If the block device contains a disklabel, and the disklabel has enough free space for a minimal installation based on the platform, the system shall allow the User to choose an automatic partition layout, to use the existing partition layout, to specify their own partition layout, or to remove the existing data and automatically label and partition the entire disk.

UI.Partition.Install.UserPrompt.LabelledAndFull. If the block device contains a disklabel, and the disklabel is either full or does not have enough free space for a minimal installation based on the platform, the system shall allow the User to choose to use the existing partition layout, to specify their own partition layout, or to remove the existing data and automatically label and partition the entire disk.

UI.Partition.Install.Automatic. If the User chooses to use an automatic disklabel and/or partition layout on the block device, the system shall create the necessary partition layout as defined by the platform of the computer.

UI.Partition.Install.Manual. If the User chooses to specify their own partition layout, the system shall allow the User to create partitions as desired.

UI.Partition.Install.Mount. If the User chooses to specify their own partition layout, or to use the existing partition layout on the disk, the system shall allow the User to choose the mount points for any extant partitions.

HorizonScript Keys

Script.Key.diskid. `diskid` — The identification strings of the disk(s).

Script.Key.disklabel. `disklabel` — Disklabels to create, if any.

Script.Key.partition. `partition` — Partitions to create, if any.

Script.Key.lvm_pv. `lvm_pv` — LVM PVs to create, if any.

Script.Key.lvm_vg. `lvm_vg` — LVM VGs to create, if any.

Script.Key.lvm_lv. `lvm_lv` — LVM LVs to create, if any.

Script.Key.encrypt. `encrypt` — Block devices to encrypt, if any.

Script.Key.fs. `fs` — File systems to create, if any.

Script.Key.mount. `mount` — File systems to mount, including block device and mountpoint.

Firmware Setup

Description / Priority

A User may possess hardware that requires non-free binary firmware to operate. This screen will allow the User to choose whether to load such firmware or not. Note that this feature can be compiled out of Horizon UI, and can also be disabled at run-time with a configuration setting.

Prerequisites

UI.Firmware.Prerequisites. The system must have firmware support compiled in.

Functional Requirements

UI.Firmware.Prompt. The system shall ask the User whether or not to load binary firmware, stressing that security-sensitive systems must not choose this option.

UI.Firmware.Load. If the User chooses to load binary firmware, the system shall add the APK Fission repository and the `linux-firmware` package to the installed environment.

HorizonScript Keys

Script.Key.firmware. `firmware` — Whether to load firmware.

Networking Setup

Description / Priority

Most Users will want to configure a network connection on their computer. The system will allow the User to configure their wired or wireless network connection, including authentication information (wireless), IP v4 and v6 addressing, routing, and DNS.

Prerequisites

UI.Network.Prerequisites. The system must have at least one non-loopback network interface detected by the kernel.

Functional Requirements

UI.Network.AddressType. The system shall ask the User whether they wish to use automatic addressing (DHCP), manual addressing, or disable network connectivity.

UI.Network.AddressType.MaybeNotDisable. The system shall not allow the User to disable network connectivity if any required repository is unavailable locally.

UI.Network.AddressType.Skip. If the User chooses to disable network connectivity, the system shall proceed to the System Metadata screen.

UI.Network.RuntimeEnv. If Horizon UI is running in a runtime environment instead of an installation environment, the system shall allow the User to add their network interface by device name before proceeding.

UI.Network.ChooseIface. If the computer has more than one network interface, the system shall prompt the User to choose which interface to use for installation.

UI.Network.Wireless. If the chosen network interface is a wireless network interface, the system shall display a Wireless Configuration screen.

UI.Network.Wireless.None. The system shall allow the User to disable wireless connectivity.

UI.Network.Wireless.None.Multi. If the User has chosen to disable wireless connectivity, and the computer has multiple network interfaces, the system shall start at UI.Network.ChooseIface again.

UI.Network.Wireless.None.Single. If the User has chosen to disable wireless connectivity, and the computer only has wireless networking, the system shall prompt the User to confirm they wish to continue without networking.

UI.Network.Wireless.APs. The system shall display a list of access points in range of the wireless interface for the User to select from.

UI.Network.Wireless.ReloadAPs. The system shall allow the User to refresh the list of access points in range.

UI.Network.Wireless.CustomAP. The system shall allow the User to input a custom access point name.

UI.Network.Wireless.Security. If the chosen access point uses authentication, the system shall prompt the User for the requisite credentials.

UI.Network.Wireless.CustomSecurity. If the User inputs a custom access point name, the system shall prompt the User for the type of security used (or none) and requisite credentials.

UI.Network.Wireless.Enable. The system shall enable the Next button when an access point is selected or input, and the requisite credentials (if any) have been input.

UI.Network.Wireless.Connect. When the User chooses Next, the system shall attempt to connect to the specified wireless network.

UI.Network.Wireless.Error. If an error occurs during wireless network connection, the system shall display a message explaning the error condition, and redisplay the Wireless Configuration screen.

UI.Network.Wireless.ConnectTimeOut. The system shall time out after waiting 15 seconds for the wireless network to connect, and consider a time out an error condition as specified in UI.Network.Wireless.Error.

UI.Network.Automatic. If the User selected automatic addressing, the system shall attempt to obtain an IPv4 address via DHCP and an IPv6 address via Router Advertisement and DHCPv6.

UI.Network.Automatic.TimeOut. If no IP address is obtained for v4 or v6 after 15 seconds, the system shall display a message that automatic configuration failed, and ask the User whether to retry the attempt, use manual addressing, or skip networking configuration.

UI.Network.WirelessPortal. If the computer has obtained an address automatically and is using a wireless network, the system shall test if the network is using a captive portal by attempting to download the file at http://detectportal.firefox.com and determining if the response is `success`.

UI.Network.WirelessPortal.Open. If the captive portal test determines a captive portal is in use, the system shall open a lightweight Web browser and attempt to navigate to http://detectportal.firefox.com/.

UI.Network.WirelessPortal.Open.Message. The system shall display a message on the lightweight Web browser that instructs the user to close the Web browser when network connectivity is established in order to continue the installation.

UI.Network.Manual. If the User selected manual address, the system shall prompt the User for the following information:

Table 3.1. Network information to collect from User for manual configuration

Information	Description
IPv4 host address	The address of this computer
IPv4 subnet mask	The subnet mask / prefix of the network
IPv4 default gateway	The router on the network
IPv4 DNS resolver	The DNS resolver to use on IPv4 networks; default to 9.9.9.9
IPv6 host address	The address of this computer
IPv6 network prefix	The prefix of the network
IPv6 default gateway	The router on the network
IPv6 DNS resolver	The DNS resolver to use on IPv6 networks; default to 2620:fe::fe

UI.Network.Manual.Test. The system shall allow the User to test their network configuration by attempting to connect to distfiles.adelielinux.org.

UI.Network.Manual.Enable. The system shall enable the Next button when either an IPv4 or IPv6 configuration is complete (address, subnet mask, gateway, DNS resolver).

HorizonScript Keys

Script.Key.network. `network` — Whether to enable network connectivity.

Script.Key.netaddress. `netaddress` — Connection information.

Script.Key.nameserver. `nameserver` — DNS resolver information.

Script.Key.netssid. `netssid` — Wireless networking information.

System Metadata Setup

Description / Priority

A User needs to specify the timezone the computer will be running in, and the computer's name. The computer's name is used even on non-networked computers, for shell prompts and login screens. The system will also allow the User to ensure the current date and time are accurate.

Functional Requirements

UI.SysMeta.DateTime. If the system is running in an installation environment, the system shall display the date and time currently stored in the RTC (Real Time Clock), and allow the User to update the date and time stored in the RTC.

UI.SysMeta.Timezone. The system shall allow the User to choose the time zone to use for time display.

UI.SysMeta.DefaultTimezone. The system shall use UTC as the default time zone.

UI.SysMeta.Hostname. The system shall allow the User to input the host name of the computer.

UI.SysMeta.DefaultHostname. The system shall provide a default host name for the User, calculated using the following algorithm:

1. If the system is running in a non-installation environment, use "Adelie" and end calculation.
2. If the system has a manufacturer available via DMI or similar API, use that name truncated to 11 characters. Otherwise, use "Adelie".
3. If the system has a network adaptor installed, even if it is not configured for use, use the last six characters of the MAC address from the first network adaptor. Otherwise, use a random fruit name.

UI.SysMeta.VerifyHostname. The system shall ensure that the host name input by the User uses only valid characters for DNS names, and not accept characters that are invalid.

UI.SysMeta.Enable. The system shall enable the Next button when the host name field contains a valid name, and disable the Next button when it is empty.

HorizonScript Keys

Script.Key.timezone. `timezone` — The zoneinfo name of the system's timezone. Ex: Africa/Nairobi.

Script.Key.hostname. `hostname` — The name of the system.

Package and Package Set Selection

Description / Priority

The system will let the User choose what packages and sets of packages to install on their computer.

Functional Requirements

UI.Packages.DisabledNetwork. If the User has chosen to disable the network, the system shall only display presets and packages available for installation locally.

UI.Packages.SimpleSel. The system shall allow the User to choose from one of the following four preset package sets, or to customise the packages installed.

UI.Packages.SimpleSel.Standard. The system shall allow the User to choose the Standard preset, which uses `adelie-base-posix` as an anchor package, and includes the `firefox-esr`, `libreoffice`, `thunderbird`, and `vlc` packages, and the `kde` and `x11` metapackages.

UI.Packages.SimpleSel.Notebook. The system shall allow the User to choose the Notebook preset, which includes the Standard preset and additional power management tools (UPower, pm-utils).

UI.Packages.SimpleSel.Minimal. The system shall allow the User to choose the Minimal preset, which uses `adelie-base` as an anchor package, and includes the `lxqt-desktop`, `featherpad`, `netsurf`, `xorg-apps`, `xorg-drivers`, and `xorg-server` packages.

UI.Packages.SimpleSel.MinimalTUI. The system shall allow the User to choose the Minimal Non-Graphical preset, which uses `adelie-base` as an anchor package, and includes the `elinks` and `tmux` packages.

UI.Packages.SimpleSel.Custom. The system shall allow the User to choose to customise the packages installed on their computer.

UI.Packages.SimpleSel.Enable. The system shall enable the Next button when a selection is made.

UI.Packages.Custom. If the User chooses Custom from UI.Packages.SimpleSel, the system shall present the User with a list of packages and package sets and allow the User to choose any or all of the packages listed. The system shall display the disk space used by each selection, and the total required for installation.

UI.Packages.Custom.Packages. The system shall offer at least the following packages, with the following names, descriptions, and corresponding APK package names.

Table 3.2. Custom packages offered on the Package Selection screen

Name	Description	APK name
Firefox Web browser	The most popular and powerful Web browser for Linux. Includes JavaScript and multimedia (audio/video) playback support.	`firefox-esr`
Netsurf Web browser	Lightweight Web browser. Does not include JavaScript support.	`netsurf`
Thunderbird Email	Read and compose email, and participate in newsgroups.	`thunderbird`
FeatherPad	Lightweight text editor.	`featherpad`
LibreOffice	Popular, extensible office suite.	`libreoffice`: `libreoffice-base`, `libreoffice-calc`, `libreoffice-draw`, `libreoffice-impress`, `libreoffice-math`, `libreoffice-writer`
VLC Media Player	Popular multimedia player, including support for a wide variety of audio and video types.	`vlc`
KDE Applications	A variety of applications including a word processor, media player, and many games.	`kde`
KDE Plasma	Modern desktop environment with graphical effect support.	`plasma-desktop`
LXQt Desktop Environment	Lightweight desktop environment using the	`lxqt-desktop`

Name	Description	APK name
	Openbox window manager and Qt widgets.	
MATE Desktop Environment	Traditional desktop environment based on the GNOME 2 look and feel.	`mate-complete`
XFCE Desktop Environment	Desktop environment based on Gtk+.	`xfce-desktop`
Window Managers	Individual window managers that are not part of a desktop environment.	`awesome`, `fluxbox`, `i3wm`, `icewm`, `openbox`, `spectrwm`

UI.Packages.Size. If the User has selected a package set or packages that will use more disk space than the computer has, the system shall prompt the User to confirm the selection and warn that installation may not be successful.

HorizonScript Keys

Script.Key.pkginstall. `pkginstall` — A space-separated list of packages to install.

Boot Setup

Description / Priority

Functional Requirements

System Accounts Setup

Description / Priority

The User needs to configure the administrator password, and provide their name/alias and password for their own account.

Functional Requirements

UI.Accounts.RootPW. The system shall prompt the User to enter a password for the root user.

UI.Accounts.RootPW.Confirm. The system shall prompt the User to re-enter the root user password to confirm accuracy.

UI.Accounts.RootPW.Explain. The system shall explain to the User the important of keeping the root user's password secure, and written in a safe place, as it is used for system administration.

UI.Accounts.RootPW.Enable. The system shall enable the Next button when the root password and confirmation match and represent a valid password.

UI.Accounts.UserAcct. The system shall prompt the User to enter their name or alias to personalise their copy of Adélie Linux, and up to four others.

UI.Accounts.UserAcct.AcctName. The system shall prompt the User to enter an account name for their user account.

UI.Accounts.UserAcct.AcctName.Default. The system shall use the following algorithm to provide a default account name for the User:

1. If the User's name or alias contains no spaces, let the account name be the entire name or alias.

2. If the User's name or alias contains spaces, let the account name be the first character of each word that is not the last word, and the full last word.

3. Transform the account name to a Latinised, lower-case form, as described by (Insert Unicode reference here).

4. If the account name is longer than 32 characters, truncate the account name to 32 characters.

UI.Accounts.UserAcct.Icon. The system shall allow the User to choose a personal icon to be shown on the system login screen from a list of preset icons.

UI.Accounts.UserAcct.Password. The system shall prompt the User to enter a password for their personal account.

UI.Accounts.UserAcct.Password.Confirm. The system shall prompt the User to re-enter their personal password to confirm accuracy.

UI.Accounts.UserAcct.Enable. The system shall enable the Next button when the user name, account name are valid, and the password and confirm password fields match and represent a valid password.

UI.Accounts.UserAcct.Others. The system shall allow the User to add up to four more user accounts, with account name, personal name/alias, personal icon, and password entry using the same requirements as the primary user acconut.

UI.Accounts.UserAcct.Others.Wheel. For each account the User adds, the system shall ask the User if the user should be an administrator or not. The primary user account will always be an administrator.

HorizonScript Keys

Script.Key.rootpw. `rootpw` — The crypt(3)-formatted, SHA-512 hashed password for the root account.

Script.Key.username. `username` — The account name(s) given.

Script.Key.useralias. `useralias` — The user's name or alias.

Script.Key.userpw. `userpw` — The crypt(3)-formatted, SHA-512 hashed password for the account(s).

Script.Key.usericon. `usericon` — Path to the chosen icon for the account(s).

Script.Key.usergroups. `usergroups` — The groups to which the user account(s) will belong. "users,lp,audio,cdrom,cdrw,scanner,camera,video,games" for normal users; add ",usb,kvm,ping,wheel" for administrators.

Save HorizonScript

Description / Priority

The User is now able to save the HorizonScript they have created using the Horizon UI. Priority: Medium, since installation environment concerns are paramount for release 1.0.

Prerequisites

UI.Writeout.Prerequisites. The system must be running in a runtime environment, not an installation environment.

Functional Requirements

UI.Writeout.Explain. The system shall inform the User that the system has finished collecting information and is now ready to save the resultant HorizonScript.

UI.Writeout.Button. The system shall replace the "Next" button with a "Save" button using the same keyboard accelerator as Next.

UI.Writeout.Save. When the User chooses "Save", the system shall open a dialogue for the user to navigate to the directory in which they wish to save the HorizonScript, and the name of the file-on disk which shall default to `installfile`.

UI.Writeout.Close. If the HorizonScript file is saved successfully, the system shall exit.

UI.Writeout.Failure. If the HorizonScript file cannot be saved successfully, the system shall display a message indicating the error, and then redisplay the writeout screen.

Begin Installation

Description / Priority

The system will describe to the User what will be done. The User may confirm, or go back.

Functional Requirements

UI.Commit.Explain. The system shall explain to the User that the next step will be performing the installation.

UI.Commit.Explain.Disk. If the disk partitioning will cause existing data to be modified or destroyed, the system shall additionally explain to the User that proceeding will modify or destroy the data on the disk(s) partitioned.

UI.Commit.Explain.Info. The system shall display the major choices the User has made in the installation. The choices shown shall be determined later during the UX design phase and are intentionally omitted from this requirements specification.

UI.Commit.Buttons. The system shall remove the Cancel button, and the Next button shall be labelled Install.

Perform Installation

Description / Priority

The system installs Adélie Linux per the User's choices.

Prerequisites

UI.Perform.Prerequisites.Wizard. If the system is running in a graphical installation environment, the HorizonScript used shall be the HorizonScript generated from the User's choices in the Horizon UI.

UI.Perform.Prerequisites.GUIServer. If the system was booted with a HorizonScript specified, and the graphical progress output option is enabled, the HorizonScript used shall be the HorizonScript loaded from the file or network location specified.

Functional Requirements

UI.Perform. The system shall execute the Horizon Runner with the specified HorizonScript, and display the installation progress.

UI.Perform.Buttons. The system shall display no buttons during installation.

UI.Perform.Status. The system shall update the current step being performed, the progress of the step being performed, and the overall progress of the installation, whenever it receives an update from the Horizon Runner.

UI.Perform.Error. If the Horizon Runner reports an error, the system shall display a screen explaining the error, possible causes, possible solutions, and that the installation has failed.

UI.Perform.Error.Save. The system shall allow the User to save the HorizonScript used, along with relevant log files, to a storage location.

UI.Perform.Complete. When Horizon Runner reports that installation is complete, the system shall continue to UI.Finish.

Installation Completion

Description / Priority

The system informs the User that installation has completed successfully, and allows the User to optionally save the HorizonScript and log files, then remove any media and reboot the computer.

Functional Requirements

UI.Finish. The system shall display a screen congratulating the User on a successful installation of Adélie Linux.

UI.Finish.Save. The system shall allow the User to save the HorizonScript and log files related to this installation to a storage location.

UI.Finish.Save.Error. If an error occurs while saving the HorizonScript and log files, the system shall display a message explaining the error, and allow the User to choose a different storage location.

UI.Finish.RemoveMedia. The system shall prompt the User to remove any installation media used, including optical media.

UI.Finish.Buttons. The system shall only display a single button: Complete, which uses the same keyboard accelerator as the Next button. The Complete button shall cause the system to reboot.

UI.Finish.Automatic. If the system is using a HorizonScript loaded from a file or network location, then the system shall automatically reboot the computer in 15 seconds.

Horizon Runner

The Horizon Runner is the component of Project Horizon that configures a computer to match an input HorizonScript.

Locate HorizonScript

Description / Priority

The system needs to load the HorizonScript into memory. If the HorizonScript is a local file, the system needs to ensure it is present and readable. If the HorizonScript is remote, the system needs to proceed to Network Configuration before continuing.

Functional Requirements

Runner.Locate.DetermineLocality. The system shall determine if the HorizonScript is stored locally or remotely using the following algorithm:

1. If the path to the HorizonScript is not provided, assume the HorizonScript is remote and end the calculation. Priority: Low.

2. If the path to the HorizonScript begins with a single "/" followed by an alphanumeric character, the HorizonScript is local.

3. If the path to the HorizonScript begins with a recognised valid protocol for downloading a HorizonScript followed by the characters "://", the HorizonScript is remote.

4. If the path to the HorizonScript begins with an alphanumeric character, the HorizonScript is local and relative to the path `/etc/horizon`.

5. If the path to the HorizonScript does not match any of the rules specified before this line, the path provided is invalid and the system shall return an error describing the specified path and the fact it is invalid.

Runner.Locate.Local. If the HorizonScript is local, the system shall ensure the HorizonScript specified exists at the specified path, is readable by the system, and is the correct format.

Runner.Locate.Local.Failure. If the HorizonScript fails the tests in Runner.Locate.Local, the system shall return an error describing the reason the HorizonScript could not be used.

Runner.Locate.Remote. If the HorizonScript is remote, the system shall configure the network and then ensure the HorizonScript specified is accessible, as described in the following requirements sections.

Runner.Locate.Remote.Configure. The system shall use the network settings configured by the Administrator on the command line, if they have been provided.

Runner.Locate.Remote.Configure.Automatic. If no network configuration was provided on the command line, the system shall initiate a DHCP request on each available network interface, waiting for a 15 second time out period before continuing to the next network interface. An

"available network interface" is defined as a non-loopback network interface that currently has a carrier signal.

Runner.Locate.Remote.Configure.AutoFailure. If no network configuration can be found using DHCP, the system shall return an error describing the inability to configure a network connection.

Runner.Locate.Remote.FullAuto. If no HorizonScript path was provided, the system shall download the HorizonScript from a TFTP server using the filename `MACADDRESS.installfile` where MACADDRESS is the MAC address of the active network adaptor with colons (:) replaced by dashes (-). The system shall use the following algorithm to determine the TFTP server, using the first match.

1. The TFTP server specified in the DHCP options, if the DHCP response provided one.
2. The TFTP server specified on the command line, if one was provided.
3. The DHCP server, if DHCP was used to configure the network.
4. The default gateway IP.

Runner.Locate.Remote.Verify. The system shall ensure that the server where the HorizonScript is kept is reachable over the protocol specified and that the HorizonScript exists.

Runner.Locate.Remote.Download. The system shall download the HorizonScript from the remote server to `/etc/horizon/`.

Validate HorizonScript

Description / Priority

The system needs to validate the supplied HorizonScript before executing it.

Functional Requirements

Runner.Validate.Exception. If the HorizonScript fails any validation step performed by the system, the system shall report the error.

Runner.Validate.Exception.Install. If the system is running in an installation environment, the system shall report validation step failure to the listening agent.

Runner.Validate.Exception.Continue. The system shall support a mode where the system will continue validation after encountering a validation failure, to display all validation failures for a given HorizonScript.

Runner.Validate.SkipBlanksAndComments. The system shall ignore any lines that are entirely blank, or start with the ASCII octothorpe (#) character.

Runner.Validate.Required. The system shall verify the presence of each required keyword in the HorizonScript: `mount`, `network`, `hostname`, `pkginstall`, and `rootpw`.

Runner.Validate.network. The system shall verify that a single `network` entry is present in the HorizonScript, and that the value is either `true` or `false`.

Runner.Validate.network.netaddress. If the value in the HorizonScript for the `network` key is `true`, the system shall verify that at least one `netaddress` key is present.

Runner.Validate.network.netaddress.NoNetwork. If the value in the HorizonScript for the `network` key is `false`, the system shall verify that no `netaddress` key is present.

Runner.Validate.network.netaddress.Validity. The system shall verify that each `netaddress` key has a valid form of at least two values in a space-separated tuple.

Runner.Validate.network.netaddress.Validity.Type. The system shall verify that the second value in each `netaddress` tuple is either `dhcp` or `static`.

Runner.Validate.network.netaddress.Validity.DHCP. The system shall verify that each `netaddress` key is a two value tuple if the second value is `dhcp`.

Runner.Validate.network.netaddress.Validity.Static. The system shall verify that each `netaddress` key is either a four value tuple or a five value tuple if the second value is `static`.

Runner.Validate.network.netaddress.Interface. If the system is running in an installation environment, the system shall ensure that the interface specified as the first value in each `netaddress` tuple is present on the system. Failure of this requirement is a "soft" error.

Runner.Validate.network.netaddress.Address. If the second value of the `netaddress` key is `static`, the system shall ensure that the third value in the tuple is a valid IPv4 or IPv6 address.

Runner.Validate.network.netaddress.Mask. If the second value of the `netaddress` key is `static`, the system shall ensure that the fourth value in the tuple is a valid prefix, in the form of a whole number between 1 and 32 inclusive if the third value is an IPv4 address, a whole number between 1 and 64 inclusive if the third value is an IPv6 address, or a network mask in the form of four octets separated by the period (`.`) symbol if the third value is an IPv4 address.

Runner.Validate.network.netaddress.Gateway. If the second value of the `netaddress` key is `static`, and the tuple contains a fifth value, the system shall ensure that the fifth value is a valid IP address of the same type as the third value in the tuple.

Runner.Validate.network.netaddress.Count. The system shall verify that `netaddress` is not specified more than 255 times per interface.

Runner.Validate.network.netssid. If the value in the HorizonScript for the `network` key is `true`, the system shall verify the validity of any present `netssid` key.

Runner.Validate.network.netssid.NoNetwork. If the value in the HorizonScript for the `network` key is `false`, the system shall verify that no `netssid` key is present.

Runner.Validate.network.netaddress.Validity. The system shall verify that each `netssid` key has a valid form of either three values or four values in a space-separated tuple.

Runner.Validate.network.netssid.Interface. If the system is running in an installation environment, the system shall ensure that the interface specified as the first value in each `netssid` tuple is present on the system and supports wireless extensions.

Runner.Validate.network.netssid.SSID. The system shall verify that the second value of each `netssid` tuple is a valid SSID enclosed in ASCII double-quotes (").

Runner.Validate.network.netssid.Security. The system shall verify that the third value of each `netssid` tuple is a valid security type: either `none`, `wep`, or `wpa`.

Runner.Validate.network.netssid.Key. If the third value of a `netssid` tuple is a valid security type and not `none`, the system shall ensure that the `netssid` tuple has a fourth value that specifies the security key.

Runner.Validate.hostname. The system shall verify that the HorizonScript contains exactly one `hostname` key.

Runner.Validate.hostname.Chars. The system shall verify that the value for the `hostname` key contains only alphanumeric and optionally one or more period (`.`) characters.

Runner.Validate.hostname.Begin. The system shall verify that the value for the `hostname` key begins with an alphabetical character.

Runner.Validate.hostname.Length. The system shall verify that the value for the `hostname` key does not exceed 320 characters in length.

Runner.Validate.pkginstall. The system shall verify that the HorizonScript contains at least one `pkginstall` key.

Runner.Validate.rootpw. The system shall verify that the HorizonScript contains exactly one `rootpw` key.

Runner.Validate.rootpw.Crypt. The system shall verify that the value for the `rootpw` key is in the format: $, either 2 for Blowfish or 6 for SHA-512, $, and then variant data.

Runner.Validate.language. The system shall verify that the HorizonScript contains zero or one `language` key.

Runner.Validate.language.Format. The system shall verify that the value of the `language` key, if present, is a valid ISO 639-1 language code, optionally followed by an ASCII underscore (_) and ISO 3166-1 country code, optionally followed by the string literal `.UTF-8`.

Runner.Validate.keymap. The system shall verify that the HorizonScript contains zero or one `keymap` key.

Runner.Validate.keymap.Valid. The system shall verify that the value of the `keymap` key, if present, is a valid keyboard layout available for use in an Adélie Linux system.

Runner.Validate.firmware. The system shall verify that the HorizonScript contains zero or one `firmware` key.

Runner.Validate.firmware.Boolean. The system shall verify that the value of the `firmware` key, if present, is either `true` or `false`.

Runner.Validate.firmware.ForceOff. If the system has firmware support compiled out, the system shall verify that the value of the `firmware` key, if present, is `false`.

Runner.Validate.timezone. The system shall verify that the HorizonScript contains zero or one `timezone` key.

Runner.Validate.timezone.zoneinfo. The system shall verify that the value of the `timezone` key, if present, represents a valid zoneinfo time zone name.

Runner.Validate.repository. The system shall verify that the HorizonScript contains zero to ten `repository` keys.

Runner.Validate.repository.ValidPath. The system shall verify that the value of each `repository` key is either an absolute local path beginning with an ASCII backslash (`/`), or a valid URL utilising the HTTP or HTTPS protocols.

Runner.Validate.signingkey. The system shall verify that the HorizonScript contains zero to ten `signingkey` keys.

Runner.Validate.signingkey.ValidPath. The system shall verify that the value of each `signingkey` key is either an absolute local path beginning with an ASCII backslash (`/`), or a valid URL utilising the HTTPS protocol.

Runner.Validate.username. The system shall verify that the HorizonScript contains zero to 255 `username` keys.

Runner.Validate.username.Unique. The system shall verify that the value of each `username` key is unique.

Runner.Validate.username.System. The system shall verify that the value of each `username` key does not match a system-defined account.

Runner.Validate.username.Valid. The system shall verify that the value of each `username` key is a valid Linux user account name.

Runner.Validate.useralias. The system shall verify that the HorizonScript contains a number of `useralias` keys equal or less than the number of `username` keys.

Runner.Validate.useralias.Validity. The system shall verify that each `useralias` key has a valid form of two values in a space-separated tuple, with the second value reading to the end of the line (optionally containing spaces).

Runner.Validate.useralias.Name. The system shall verify that the first value in each `useralias` key tuple is an account name specified in a `username` key.

Runner.Validate.useralias.Unique. The system shall verify that only one `useralias` key is specified per account name.

Runner.Validate.userpw. The system shall verify that the HorizonScript contains a number of `userpw` keys equal or less than the number of `username` keys.

Runner.Validate.userpw.Validity. The system shall verify that each `userpw` key has a valid form of two values in a space-separated tuple.

Runner.Validate.userpw.Name. The system shall verify that the first value in each `userpw` key tuple is an account name specified in a `username` key.

Runner.Validate.userpw.Unique. The system shall verify that only one `userpw` key is specified per account name.

Runner.Validate.userpw.Crypt. The system shall verify that the second value for each `userpw` key tuple is in the format: $, either 2 for Blowfish or 6 for SHA-512, $, and then variant data.

Runner.Validate.usericon. The system shall verify that the HorizonScript contains a number of `usericon` keys equal or less than the number of `username` keys.

Runner.Validate.usericon.Validity. The system shall verify that each `usericon` key has a valid form of two values in a space-separated tuple.

Runner.Validate.usericon.Name. The system shall verify that the first value in each `usericon` key tuple is an account name specified in a `username` key.

Runner.Validate.usericon.Unique. The system shall verify that only one `usericon` key is specified per account name.

Runner.Validate.usericon.ValidPath. The system shall verify that the second value of each `usericon` key tuple is either an absolute local path beginning with an ASCII backslash (`/`), or a valid URL utilising the HTTP or HTTPS protocols.

Runner.Validate.usergroups. The system shall verify any `usergroups` keys contained in the HorizonScript.

Runner.Validate.usergroups.Validity. The system shall verify that each `usergroups` key has a valid form of two values in a space-separated tuple.

Runner.Validate.usergroups.Name. The system shall verify that the first value in each `usergroups` key tuple is an account name specified in a `username` key.

Runner.Validate.usergroups.Count. The system shall verify that all `usergroups` key tuples for a specified account name specify a combined total of sixteen or fewer groups.

Runner.Validate.usergroups.Unique. The system shall verify that a group is specified only once for each account name.

Runner.Validate.usergroups.Group. The system shall verify that each group specified is a valid system-defined group name.

Runner.Validate.diskid. The system shall verify any `diskid` keys contained in the HorizonScript.

Runner.Validate.diskid.Validity. The system shall verify that each `diskid` key has a valid form of two values in a space-separated tuple, with the second value reading to the end of the line (optionally containing spaces).

Runner.Validate.diskid.Unique. The system shall verify that the first value of each `diskid` key tuple is unique in the HorizonScript.

Runner.Validate.diskid.Block. If the system is running in an installation environment, the system shall ensure that the first value of each `diskid` key tuple specifies a valid block device.

Runner.Validate.disklabel. The system shall verify any `disklabel` keys contained in the HorizonScript.

Runner.Validate.disklabel.Validity. The system shall verify that each `disklabel` key has a valid form of two values in a space-separated tuple.

Runner.Validate.disklabel.Unique. The system shall verify that the first value of each `disklabel` key tuple is unique in the HorizonScript.

Runner.Validate.disklabel.Block. If the system is running in an installation environment, the system shall ensure that the first value of each `disklabel` key tuple specifies a valid block device.

Runner.Validate.disklabel.LabelType. The system shall verify that the second value of each `disklabel` key tuple is one of the following three values: `apm`, `mbr`, or `gpt`.

Runner.Validate.partition. The system shall verify any `partition` keys contained in the HorizonScript.

Runner.Validate.partition.Validity. The system shall verify that each `partition` key has a valid form of three or four values in a space-separated tuple.

Runner.Validate.partition.Validity.PartNo. The system shall verify that the second value of each `partition` key tuple is a valid positive whole integer.

Runner.Validate.partition.Unique. The system shall verify that the union of the first and second values in each `partition` key tuple are unique.

Runner.Validate.partition.Block. If the system is running in an installation environment, the system shall ensure that the first value of each `partition` key tuple specifies a valid block device.

Runner.Validate.partition.Size. The system shall verify that the third value in each `partition` key tuple specifies a valid size as described in HorizonScript Specification, 2019, Wilcox et al.

Runner.Validate.partition.TypeCode. If a `partition` key tuple contains a fourth value, the system shall verify that the fourth value of the `partition` key tuple is a valid type code, which is one of the following two values: `boot`, or `esp`.

Runner.Validate.lvm_pv. The system shall verify any `lvm_pv` keys contained in the HorizonScript.

Runner.Validate.lvm_pv.Validity. The system shall verify that the value of each `lvm_pv` is an absolute path to a device node.

Runner.Validate.lvm_pv.Block. If the system is running in an installation environment, the system shall verify that the value of each `lvm_pv` key specifies a valid block device.

Runner.Validate.lvm_vg. The system shall verify any `lvm_vg` keys contained in the HorizonScript.

Runner.Validate.lvm_vg.Validity. The system shall verify that each `lvm_vg` key has a valid form of two values in a space-separated tuple.

Runner.Validate.lvm_vg.Block. If the system is running in an installation environment, the system shall verify that the first value of each `lvm_vg` key tuple specifies a valid block device.

Runner.Validate.lvm_vg.Name. The system shall verify that the second value of each `lvm_vg` key specifies a string that would be valid as an LVM volume group name.

Runner.Validate.lvm_lv. The system shall verify any `lvm_lv` keys contained in the HorizonScript.

Runner.Validate.lvm_lv.Validity. The system shall verify that each `lvm_lv` key has a valid form of three values in a space-separated tuple.

Runner.Validate.lvm_lv.VolumeGroup. If the system is running in an installation environment, the system shall verify that the first value of each `lvm_lv` key specifies a valid LVM volume group, or one specified in an `lvm_vg` key.

Runner.Validate.lvm_lv.Name. The system shall verify that the second value of each `lvm_lv` key specifies a string that would be valid as an LVM logical volume name.

Runner.Validate.lvm_lv.Size. The system shall verify that the third value in each `lvm_lv` key tuple specifies a valid size as described in HorizonScript Specification, 2019, Wilcox et al.

Runner.Validate.encrypt. The system shall verify any `encrypt` keys contained in the HorizonScript.

Runner.Validate.encrypt.Validity. The system shall verify that the value of each `encrypt` key is a string value that is an absolute path to a device node, with an optional second value as a space-separated tuple.

Runner.Validate.encrypt.Block. If the system is running in an installation environment, the system shall verify that the value of each `encrypt` key, and the first value of each `encrypt` key tuple, specifies a valid block device.

Runner.Validate.fs. The systemm shall verify any `fs` keys contained in the HorizonScript.

Runner.Validate.fs.Validity. The system shall verify that each `fs` key has a valid form of two values in a space-separated tuple.

Runner.Validate.fs.Block. If the system is running in an installation environment, the system shall verify that the first value of each `fs` key tuple specifies a valid block device.

Runner.Validate.fs.Type. The system shall verify that the second value of each `fs` key tuple specifies a valid file system supported by Horizon, using lower case characters.

Runner.Validate.mount. The system shall verify that the HorizonScript contains at least one valid `mount` key.

Runner.Validate.mount.Validity. The system shall verify that each `mount` key has a valid form of two or three values in a space-separated tuple.

Runner.Validate.mount.Block. If the system is running in an installation environment, the system shall verify that the first value of each `mount` key tuple specifies a valid block device.

Runner.Validate.mount.Point. The system shall verify that the second value of each `mount` key specifies a valid mount point beginning with a `/`.

Runner.Validate.PackageAvail. If the system is running in a runtime environment, the system shall verify that all of the packages specified in `pkginstall` keys are available in the specified repositories; either the repositories specified in `repository` keys, or the system-default repositories if no `repository` keys have been specified.

Execute HorizonScript

Description / Priority

The system executes the instructions provided in the HorizonScript, in the specified order. The system reports progress, and all warnings and errors, to the listening agent. Once the system has finished, it reports the completion status to the listening agent.

Functional Requirements

Runner.Execute. The system shall execute the instructions provided in the HorizonScript in the order described.

Runner.Execute.Exception. The system shall report any exception encountered during execution to the listening agent, following the specification in the section called “Runner Listening Agent Interface”.

Runner.Execute.Status. If an operation has taken more than 10 seconds, the system shall report the status of the operation to the listening agent, following the specification in the section called “Runner Listening Agent Interface”.

Runner.Execute.StepBegin. When the system begins a step, the system shall report the step as beginning to the listening agent, following the specification in the section called “Runner Listening Agent Interface”.

Runner.Execute.StepDone. When the system completes a step, the system shall report the step as completed to the listening agent, following the specification in the section called “Runner Listening Agent Interface”.

Runner.Execute.Verify. The system shall validate the HorizonScript, using the process specified in the section the section called “Validate HorizonScript”, before execution.

Runner.Execute.Verify.Failure. If the HorizonScript fails validation, the system shall report a fatal exception (Runner.Execute.Exception) and stop execution.

Runner.Execute.diskid. The system shall verify that each block device specified in a `diskid` key tuple matches the identification string provided.

Runner.Execute.diskid.Failure. If any block device fails the `diskid` identification string check, the system shall consider this a fatal error and stop execution of the HorizonScript.

Runner.Execute.disklabel. The system shall create any disklabels specified in `disklabel` key tuples.

Runner.Execute.disklabel.Overwrite. If a disklabel already exists on the block device where a new disklabel will be written, the system shall erase the existing disklabel before creating the new disklabel.

Runner.Execute.disklabel.Failure. If a disklabel cannot be created due to an I/O error, or due to a non-existent block device, the system shall consider this a fatal error and stop execution of the HorizonScript.

Runner.Execute.partition. The system shall create any partitions specified in `partition` key tuples.

Runner.Execute.partition.Failure. The system shall consider any failure to create a partition as a fatal error and stop execution of the HorizonScript, unless otherwise specified by a requirement in this tree.

Runner.Execute.partition.Failure.Device. If a partition cannot be created due to an I/O error, or due to a non-existent block device, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.partition.Failure.Numbering. The system shall consider a numbering error, defined as creating a partition with the same index as an existing partition or creating a partition with an invalid index for the block device's label, a fatal error and stop execution of the HorizonScript.

Runner.Execute.partition.Failure.Duplicate. If the partition cannot be created because it already exists, and all properties are identical (index, size, type codes), the system shall continue as if the partitioning succeeded.

Runner.Execute.encrypt.PVs. If any `lvm_pv` keys are specified, and any `encrypt` keys specify the block device where an LVM physical volume will be created, the system shall execute the `encrypt` keys that correlate with `lvm_pv` keys before executing the `lvm_pv` keys.

Runner.Execute.lvm_pv. The system shall create any LVM physical volumes specified in `lvm_pv` keys.

Runner.Execute.lvm_pv.Failure. If a LVM physical volume cannot be created due to an I/O error, or due to a non-existent block device, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.lvm_vg. The system shall create any LVM volume groups specified in `lvm_vg` keys.

Runner.Execute.lvm_vg.Duplicate. If an LVM volume group with the specified name already exists on the computer, the system shall consider this condition a fatal error and stop execution of the HorizonScript unless the volume group is using the same physical volume as specified, in which case the system shall continue as if creation of the LVM volume group succeeded.

Runner.Execute.lvm_vg.Failure. If an LVM volume group cannot be created due to an I/O error, or due to a non-existent physical volume, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.encrypt.LVs. If any `lvm_lv` keys are specified, and any `encrypt` keys specify a LVM logical volume that will be created, the system shall execute the `encrypt` keys that correlate with `lvm_lv` keys before executing the `lvm_lv` keys.

Runner.Execute.lvm_lv. The system shall create any LVM logical volumes specified in `lvm_lv` keys.

Runner.Execute.lvm_lv.Failure. If an LVM logical volume cannot be created due to an I/O error, or due to a non-existent volume group, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.encrypt. The system shall create LUKS containers specified in `encrypt` keys.

Runner.Execute.encrypt.Prompt. If an `encrypt` key does not specify a passphrase, the system shall prompt the User for a passphrase via the listening agent, following the specification in the section called “Runner Listening Agent Interface”.

Runner.Execute.fs. The system shall create file systems specified in `fs` key tuples.

Runner.Execute.fs.Overwrite. If a file system is present on the block device where a new file system is to be created, the system shall overwrite the existing file system.

Runner.Execute.fs.Failure. If a file system cannot be created due to an I/O error, or due to a non-existent block device, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.mount. The system shall mount the file systems specified in `mount` keys, under the `/target` namespace.

Runner.Execute.mount.UnreadableFS. If a block device specified in a `mount` key tuple has a file system that is unreadable by the computer, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.mount.InvalidOption. If a mount option specified in a `mount` key tuple is invalid, and the file system successfully mounts without that option set, the system shall log a warning to the listening agent and proceed with installation.

Runner.Execute.mount.Failure. If a file system cannot be mounted due to an I/O error, or due to a non-existent block device, the system shall consider this condition a fatal error and stop execution of the HorizonScript.

Runner.Execute.hostname. The system shall set the hostname of the computer to the hostname specified in the `hostname` key.

Runner.Execute.hostname.Write. The system shall write the hostname of the computer to the `/etc/hostname` file in the target namespace.

Runner.Execute.rootpw. The system shall set the root password in the target namespace to the value specified in the `rootpw` key.

Runner.Execute.language. If a `language` key is specified in the HorizonScript, the system shall add a script to `/etc/profile.d` setting the `LANG` environment variable to the value of the `language` key.

Runner.Execute.keymap. If a `keymap` key is specified in the HorizonScript, the system shall set the `XKBLAYOUT` variable in the `/etc/default/keyboard` file in the target namespace to value of the `keymap` key.

Runner.Execute.UserAccounts. The system shall configure user accounts in the target namespace as specified in keys `username`, `useralias`, `userpw`, and `usergroups`.

Runner.Execute.repository. If one or more `repository` keys are specified in the HorizonScript, the system shall write the repository locations to the `/etc/apk/repositories` file in the target namespace; otherwise, the system shall write the default repository locations to the `/etc/apk/repositories` file in the target namespace.

Runner.Execute.firmware. If a `firmware` key is specified in the HorizonScript, and the system is compiled with firmware support enabled, and the value of the `firmware` key is `true`, the system shall add `linux-firmware` to the list of packages to install.

Runner.Execute.firmware.Repository. If no `repository` keys are specified in the HorizonScript, the system shall append the repository `https://distfiles.apkfission.net/adelie-$VERSION/nonfree` to the `/etc/apk/repositories` file in the target namespace, where `$VERSION` is the version of Adélie Linux being installed.

Runner.Execute.netssid. The system shall write the wireless networking configuration specified by any `netssid` keys to the `/etc/wpa_supplicant/wpa_supplicant.conf` file in the target namespace.

Runner.Execute.netaddress. The system shall write the networking configuration specified by any `netaddress` keys to the `/etc/conf.d/net` file in the target namespace.

Runner.Execute.netaddress.OpenRC. This requirement is under discussion.

Runner.Execute.nameserver. If one or more `nameserver` keys are specified in the HorizonScript, the system shall write the nameservers specified to the `/etc/resolv.conf` file in the target namespace.

Runner.Execute.network. If the value of the `network` key is `true`, the system shall execute the `netssid`, `netaddress`, and `nameserver` keys in the installation environment.

Runner.Execute.network.netaddress. The system shall use the networking configuration specified in any `netaddress` keys in the installation environment.

Runner.Execute.network.netssid. If the value of the `network` key is `true`, the system shall use the wireless networking configuration specified in any `netssid` keys in the installation environment.

Runner.Execute.network.nameserver. If the value of the `network` key is `true`, the system shall use the nameservers specified in any `nameserver` keys in the installation environment.

Runner.Execute.usericon. The system shall use the icons specified in any `usericon` key tuples for identification of users in user interface elements in the target namespace.

Runner.Execute.usericon.Remote. If any `usericon` location is remote, the system shall download the contents of the remote file to the `/usr/share/user-manager/avatars/remote/` directory in the target namespace.

Runner.Execute.signingkey. If any `signingkey` key is specified in the HorizonScript, the system shall copy all specified files to the `/etc/apk/keys/` directory in the target namespace; otherwise, the system shall copy the default Adélie Linux signing keys to the `/etc/apk/keys/` directory in the target namespace.

Runner.Execute.signingkey.Firmware. If the `firmware` key is specified in the HorizonScript, and the system is compiled with firmware support enabled, and the value of the `firmware` key is `true`, then the system shall copy the APK Fission signing key(s) to the `/etc/apk/keys/` directory in the target namespace.

Runner.Execute.pkginstall. The system shall install all packages requested via the `pkginstall` key to the target namespace utilising the repositories configured in the target namespace.

Runner.Execute.pkginstall.APKDB. The system shall initialise the APK database in the target namespace before installing packages.

Runner.Execute.timezone. If a `timezone` key is specified in the HorizonScript, the system shall set the timezone in the target namespace to the specified timezone; otherwise, the system shall set the timezone in the target namespace to UTC.

Runner.Execute.timezone.Missing. If the zoneinfo file for the specified timezone is missing in the target namespace, but present in the installation environment, the system shall copy the zoneinfo file from the installation environment to the `/etc/localtime` file in the target namespace.

Runner.Execute.Finish. The system shall notify the listening agent that the HorizonScript has executed successfully and that the target is installed.

Chapter 4. External Interface Requirements

User Interfaces

Iface.UI.IconTheme. The system shall use the Papirus icon set for any icons displayed in the user interface. Priority: Medium.

Iface.UI.TabStops. The system shall ensure that each interactive control be assigned a tab stop in order from top left to top right to bottom left to bottom right.

Iface.UI.Accelerators. The system shall ensure that interactive controls may be selected for input via a keyboard accelerator. The keyboard accelerator shall be in the form of Alt+[key], and the key shall be discoverable by underlining the letter corresponding to the key in the label for the interactive control.

Iface.UI.ButtonAccel. The system shall ensure that all displayed buttons have a corresponding function key as a keyboard accelerator. The function key shall be displayed as a parenthetical note next to the label of the button. For example: "Next (F6)", "Cancel (F3)".

Iface.UI.ScreenRes. The system shall be usable with a minimum screen resolution of 800x600 pixels.

Iface.UI.Scaling. The system shall ensure that controls and fonts are scaled based on the DPI of the screen.

Iface.UI.StandardButtons. Horizon UI shall have a "Back (F5)", "Next (F8)", "Cancel (F3)", and "Help (F1)" on all pages except the first page (where the Back button shall not be displayed) and the last page (where the Back and Cancel buttons shall not be displayed).

Iface.UI.ScreenReader. The system shall be usable with the built-in Qt screen reader. Priority: Low.

Iface.UI.Colours. The system shall not rely on colour alone to differentiate options, controls, or other elements. The system shall additionally be tested in a greyscale mode.

Iface.UI.NeutralColours. The system shall use neutral colours for the window title bar, window border, and background.

Hardware Interfaces

No direct hardware interfaces have been identified for this release of Project Horizon; all interfacing with hardware is done via other libraries as defined below in the section called “External Software Interfaces”.

External Software Interfaces

Iface.Software.WPA. The system shall use the wpactrl library [https://www.skarnet.org/software/bcnm/libwpactrl/] from bcnm [https://www.skarnet.org/software/bcnm/] for manipulating wireless network connections.

Iface.Software.blkid. The system shall use the blkid library [https://linux.die.net/man/3/libblkid] from util-linux [https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git] for determining current disk layout.

Iface.Software.Parted. The system shall use the libparted library from parted [https://www.gnu.org/software/parted/] for manipulating disk partitions.

Iface.Software.dhcpcd. The system shall use dhcpcd [https://roy.marples.name/projects/dhcpcd] for automatically obtaining network configuration over the DHCP protocol.

Iface.Software.Eudev. The system shall use the libudev library [https://www.freedesktop.org/software/systemd/man/libudev.html] from eudev [https://wiki.gentoo.org/wiki/Eudev] for enumerating devices present on the system, including but not limited to: disk drives, network interfaces, graphics adaptors.

Iface.Software.Timezones. The system shall use tzdata for time zone information and selection, and shall write the selected time zone to `/etc/localtime` in the installation environment.

External Communication Interfaces

Iface.Comm.Serial. Horizon Runner shall support writing progress and log messages to a serial interface if requested by the user.

Iface.Comm.UI.UploadLog. Horizon UI shall support uploading of log files to an external service in the event of an installation failure.

Iface.Comm.UI.Network.Wireless.SecTypes. The system shall support the following security types for wireless networking access points:

1. No Security
2. WEP Passphrase
3. WPA with Preshared Key (PSK)

Runner Listening Agent Interface

The Horizon Runner Listening Agent Interface is the interface used to communicate progress. The Horizon Runner will send messages to the Listening Agent using the interface defined here to note status messages, step completion, and any exceptions that occur during execution. The Listening Agent may be an external application (such as the Horizon UI), a logging system, or even a serial port or TTY monitored by the user. For this reason, automated parsing is a higher priority than sapient parsing, but sapient parsing still must be considered.

Agent.MessageFormat. Messages sent by the Agent shall be in the following format:

1. A timestamp, in ISO 8601 format;
2. A tab character (`\t`);
3. Message class: `log`, `prompt`, `stepstart`, `stepend`
4. A tab character (`\t`);

5. The message;

6. A newline character (\n).

Agent.Prompt. When the Agent requires external input from the User, the system shall interpret the answer as an entire line sent to the Agent via STDIN terminated with a newline.

Chapter 5. Non-functional Requirements

Performance Requirements

For the purposes of this section, the base hardware platforms shall be defined as follows.

Base hardware platform, PowerPC

CPU	PowerPC "G3" 750 at 600 MHz
RAM	640 MB PC133
Disk drive	40 GB PATA
Optical drive	8X DVD-ROM drive

Base hardware platform, Intel

CPU	Pentium III at 550 MHz
RAM	256 MB PC133
Disk drive	80 GB PATA
Optical drive	24X CD-ROM drive

Performance.BootToX

Ambition	Fast boot times for the installation environment.
Scale	Seconds elapsed between the kernel being loaded by GRUB and the display of the initial Horizon UI window accepting input.
Meter	Stopwatch testing performed five times on each base hardware platform. Does not include UI.Early.MultiGfx.
Must	70 seconds.
Plan	60 seconds.
Wish	45 seconds.

Performance.ScreenChange

Ambition	Responsiveness for screen changes in the Horizon UI.
Scale	Milliseconds elapsed between choosing the "Next" button in the Horizon UI and the next screen being displayed and able to accept input.
Meter	Printed statements to the stderr log including a timestamp when the system is aware of the Next button being pressed and when the system has fully loaded the next screen and is accepting input. Test each screen through a complete installation flow on each base hardware platform ten times.

Must 1000 milliseconds for the worst case.

Plan 750 milliseconds for the worst case.

Wish 300 milliseconds for the worst case.

Performance.Validation

Ambition Fast validation of HorizonScript files.

Scale Seconds elapsed between the start and completion of a validation job. Test validation of "typical" installation scripts on each base hardware platform 50 times. Do not include network resources in installation scripts.

Meter Shell **time** built-in running the validation tool.

Must 5 seconds.

Plan 3 seconds.

Wish 1 second or less.

Safety Requirements

Safety.DiskLock.UI. The Horizon UI system shall not write any data to any block device, except to save log files and/or an HorizonScript file explicitly requested to be saved by the User.

Safety.DiskLock.Runner. The Horizon Runner system shall not write any data to any block device until the HorizonScript is fully validated.

Security Requirements

Security.PasswordHash. The Horizon UI system shall store passwords in memory in their hashed versions only, and securely clear the memory storage that contained plain-text passwords once the hash is stored.

Software Quality Attributes

SQA.Flexibility.ScriptFormat. A programmer with two years of C++ experience shall be able to change the on-disk format of HorizonScript with no more than two hours of labour.

SQA.Flexibility.Firmware. The firmware support described under the UI.Firmware.* requirements branch shall be disableable at compile-time.

SQA.Flexibility.FirmwareConf. The firmware support described under the UI.Firmware.* requirements branch shall be disableable at run-time via a configuration setting.

SQA.Robustness.ScriptSync. The system shall provide a method to save the executed HorizonScript and log messages in the event of a fatal error.

SQA.Usability.TTI.Experienced. A User with experience installing other Linux distributions shall be able to complete the Horizon UI flow from language selection through the section called “Perform Installation” in less than 15 minutes.

SQA.Usability.TTI.Green. A User with no prior experience with Linux shall be able to complete the Horizon UI flow from language selection through the section called “Perform Installation” in less than one hour.

SQA.Portability.Bitness. A HorizonScript shall be readable on 32-bit and 64-bit computers regardless of what type of computer was used to create it.

SQA.Portability.Endianness. A HorizonScript shall be readable on big endian and little endian computers regardless of what type of computer was used to create it.

SQA.Verifiability.BlockMock. The system shall support the ability to use "mock" block devices for purposes of testing and verification.

SQA.Maintainability.Comments. All functions and methods in the system shall have a comment describing what action the function or method performs, and its inputs, outputs, and any side effects (if applicable).

SQA.Maintainability.Methods. Each class in the system shall have no more than 20 methods.

SQA.Maintainability.NoOPFuncs. Each function or method in the system shall take no more than six input parameters.

Chapter 6. Other Requirements

This chapter describes requirements that apply to the system but do not fit cleanly in other chapters.

User Documentation

UserDoc.JargonFree. The system documentation must contain minimal uses of technical jargon. The system documentation must define technical jargon where it is first used. "Jargon" is defined as a word that is technical in nature and is not listed in a Webster's or Oxford English dictionary.

UserDoc.ContextAware. The system must allow the user to read documentation relevant to the phase of installation that is currently on-screen.

Logging Requirements

Logging.Levels. The system shall have four levels of logging: Error, Warning, Information, Debug.

Logging.Levels.Tiers. The system shall treat each level of logging as containing the prior level: for example, a level of Warning will additionally contain all Error output.

Logging.Levels.Default. The system shall default to a logging level of Information unless a different logging level is chosen during invocation.

Logging.ExtProcess. The system shall log all processes executed, including binary path, arguments, and environment, in the Debug logging level.

Logging.ExtProcess.ReturnCode. The system shall log all external process return codes, including name of binary and brief description of purpose of execution, in the Information log level.

Logging.ExtProcess.ReturnCode.Error. If the return code of an external process is non-zero, the system shall log the external process return code message in the Error log level.

Logging.UIKeys. The Horizon UI system shall log the values it uses for each key as it gathers information in the Debug logging level.

Internationalisation Requirements

I18n.Meow. The Horizon UI system shall be implemented in English and at least one fake language for the purposes of ensuring that all strings are translatable.