LiveCode 9.6.8 Release Notes

Overview

This document describes all the changes that have been made for LiveCode 9.6.8, including bug fixes and new syntax.

Platform support

The engine supports a variety of operating systems and versions. This section describes the platforms that we ensure the engine runs on without issue (although in some cases with reduced functionality).

Windows

LiveCode supports the following versions of Windows:

- Windows 7 (both 32-bit and 64-bit)
- Windows Server 2008
- Windows 8.x (Desktop)
- Windows 10
- Windows 11

Note: On 64-bit Windows installations, LiveCode can run either as a 32-bit application through the WoW layer or as a native 64-bit Windows application, depending on the installer that is chosen.

Linux

LiveCode supports the following Linux distributions, on 32-bit or 64-bit Intel/AMD or compatible processors:

- Ubuntu 14.04 and 16.04
- Fedora 23 & 24
- Debian 7 (Wheezy) and 8 (Jessie) [server]
- CentOS 7 [server]

LiveCode may also run on Linux installations which meet the following requirements:

- Required dependencies for core functionality:
  - glibc 2.13 or later
  - glib 2.0 or later
- Optional requirements for GUI functionality:
  - GTK/GDK 2.24 or later
  - Pango with Xft support
  - esd (optional, needed for audio output)
  - mplayer (optional, needed for media player functionality)
  - lcms (optional, required for color profile support in images)
  - gksu (optional, required for privilege elevation support)

Note: If the optional requirements are not present then LiveCode will still run but the specified features will be disabled.

Note: The requirements for GUI functionality are also required by Firefox and Chrome, so if your Linux distribution runs one of those, it will run LiveCode.

macOS

The macOS engine supports Intel architecture machines running the following operating system versions:

- 10.9.x (Mavericks)
- 10.10.x (Yosemite)
- 10.11.x (El Capitan)
- 10.12.x (Sierra)
- 10.13.x (High Sierra)
- 10.14.x (Mojave)
- 10.15.x (Catalina)
- 11.x (Big Sur)
- 12.x (Monterey)

The macOS engine supports Apple architecture machines running the following operating system versions:

- 11.x (Big Sur)
- 12.x (Monterey)

The macOS IDE supports building macOS standalones with an Apple architecture slice on:

- 10.13.x (High Sierra)
- 10.14.x (Mojave)
- 10.15.x (Catalina)
- 11.x (Big Sur)
- 12.x (Monterey)

When running the IDE in older macOS versions, the request for adding an Apple architecture slice will be ignored.

Note: Apple architecture support is currently experimental. To run the IDE using Apple architecture (on supported machines), you must toggle the `Open using Rosetta` option to off in the `LiveCode.app` bundle's `Get Info` pane in Finder. To build a standalone with native Apple architecture support you must explicitly choose the `macOS Apple` option in standalone settings.

iOS

iOS deployment is possible when running LiveCode IDE on a Mac, and provided Xcode is installed and has been set in LiveCode *Preferences* (in the *Mobile Support* pane).

Currently, the supported versions of Xcode are:

- Xcode 10.1 on MacOS 10.13 (Note: You need to upgrade to 10.13.4)
- Xcode 11.3 on MacOS 10.14 (Note: You need to upgrade to 10.14.4)
- Xcode 12.4 on MacOS 10.15 and above (Note: You need to upgrade to 10.15.4)
- Xcode 13.2 on MacOS 11 and above (Note: You need to upgrade to 11.3+)

It is also possible to set other versions of Xcode, to allow testing on a wider range of iOS simulators. For instance, on MacOS 10.14 (High Sierra), you can add *Xcode 10.1* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 12.1*.

We currently support building against the following versions of the iOS SDK:

- 12.1 (included in Xcode 10.1)
- 13.2 (included in Xcode 11.3)
- 14.4 (included in Xcode 12.4)

- 15.2 (included in Xcode 13.2)

Android

LiveCode allows you to save your stack as an Android application, and also to deploy it on an Android device or simulator from the IDE.

Android deployment is possible from Windows, Linux and Mac OSX.

The Android engine supports devices using x86, x86-64, ARM and ARM64 processors. It will run on the following versions of Android:

- 5.0-5.1 (Lollipop)
- 6.0 (Marshmallow)
- 7.x (Nougat)
- 8.x (Oreo)
- 9.0 (Pie)
- 10.0 (Q)
- 11.0 (R)

To enable deployment to Android devices, you need to download the Android SDK, and then use the 'Android SDK Manager' to install:

- the latest "Android SDK Tools"
- the latest "Android SDK Platform Tools"

You also need to install the Java Development Kit (JDK). On Linux, this usually packaged as "openjdk". LiveCode requires JDK version 1.6 or later.

Once you have set the path of your Android SDK in the "Mobile Support" section of the LiveCode IDE's preferences, you can deploy your stack to Android devices.

Some users have reported successful Android Watch deployment, but it is not officially supported.

HTML5

LiveCode applications can be deployed to run in a web browser, by running the LiveCode engine in JavaScript and using modern HTML5 JavaScript APIs.

HTML5 deployment does not require any additional development tools to be installed.

LiveCode HTML5 standalone applications are currently supported for running in recent versions of Mozilla Firefox, Google Chrome or Safari. For more information, please see the "HTML5 Deployment" guide in the LiveCode IDE.

Setup

Installation

Each version of LiveCode installs can be installed to its own, separate folder. This allow multiple versions of LiveCode to be installed side-by-side. On Windows (and Linux), each version of LiveCode has its own Start Menu (or application menu) entry. On Mac OS X, each version has its own app bundle.

On Mac OS X, install LiveCode by mounting the `.dmg` file and dragging the app bundle to the `Applications` folder (or any other suitable location).

For Windows and Linux, the default installation locations when installing for "All Users" are:

Platform	Path
Windows	`<x86 program files folder>/RunRev/LiveCode <version>`
Linux	`/opt/livecode/livecode-<version>`

The installations when installing for "This User" are:

Platform	Path
Windows	`<user roaming app data folder>/RunRev/Components/LiveCode <version>`
Linux	`~/.runrev/components/livecode-<version>`

Note: If installing for "All Users" on Linux, either the **gksu** tool must be available, or you must manually run the LiveCode installer executable as root (e.g. using **sudo** or **su**).

Uninstallation

On Windows, the installer hooks into the standard Windows uninstall mechanism. This is accessible from the "Add or Remove Programs" applet in the windows Control Panel.

On Mac OS X, drag the app bundle to the Trash.

On Linux, LiveCode can be removed using the `setup.x86` or `setup.x86_64` program located in LiveCode's installation directory.

Reporting installer issues

If you find that the installer fails to work for you then please report it using the LiveCode Quality Control Centre or by emailing support@livecode.com.

Please include the following information in your report:

- Your platform and operating system version
- The location of your home or user folder
- The type of user account you are using (guest, restricted, admin etc.)
- The installer log file.

The installer log file can be located as follows:

Platform	Path

Platform	Path
Windows 2000/XP	`<documents and settings folder>/<user>/Local Settings/`
Windows Vista/7	`<users folder>/<user>/AppData/Local/RunRev/Logs`
Linux	`<home>/.runrev/logs`

Activating LiveCode

The licensing system ties your product licenses to a customer account system, meaning that you no longer have to worry about finding a license key after installing a new copy of LiveCode. Instead, you simply have to enter your email address and password that has been registered with our customer account system and your license key will be retrieved automatically.

Alternatively it is possible to activate the product via the use of a specially encrypted license file. These will be available for download from the customer center after logging into your account. This method will allow the product to be installed on machines that do not have access to the internet.

Command-line installation

It is possible to invoke the installer from the command-line on Linux and Windows. When doing command-line installation, no GUI will be displayed. The installation process is controlled by arguments passed to the installer.

Run the installer using a command in the form:

```
<installer> install -ui [OPTION ...]
```

where `<installer>` should be replaced with the path of the installer executable or app (inside the DMG) that has been downloaded. The result of the installation operation will be written to the console.

The installer understands any of the following `OPTION`s:

Option	Description
`-allusers`	Install the IDE for "All Users". If not specified, LiveCode will be installed for the current user only.
`-desktopshortcut`	Place a shortcut on the Desktop (Windows-only)
`-startmenu`	Place shortcuts in the Start Menu (Windows-only)
`-location LOCATION`	The folder to install into. If not specified, the `LOCATION` defaults to those described in the "Installation" section above.
`-log LOGFILE`	The file to which to log installation actions. If not specified, no log is generated.

Note: the command-line installer does not do any authentication. When installing for "All Users", you will need to run the installer command as an administrator.

As the installer is actually a GUI application, it needs to be run slightly differently from other command-line programs.

On Windows, the command is:

```
start /wait <installer> install -ui [OPTION ...]
```

Command-line uninstallation

It is possible to uninstall LiveCode from the command-line on Windows and Linux. When doing command-line uninstallation, no GUI will be displayed.

Run the uninstaller using a command of the form:

```
<uninstaller> uninstall -ui
```

Where is *.setup.exe* on Windows, and *.setup.x86* on Linux. This executable, for both of the platforms, is located in the folder where LiveCode is installed.

The result of the uninstallation operation will be written to the console.

Note: the command-line uninstaller does not do any authentication. When removing a version of LiveCode installed for "All Users", you will need to run the uninstaller command as an administrator.

Command-line activation

It is possible to activate an installation of LiveCode for all users by using the command-line. When performing command-line activation, no GUI is displayed. Activation is controlled by passing command-line arguments to LiveCode.

Activate LiveCode using a command of the form:

```
<livecode> activate -file LICENSEFILE -passphrase SECRET
```

where `<livecode>` should be replaced with the path to the LiveCode executable or app that has been previously installed.

This loads license information from the manual activation file `LICENSEFILE`, decrypts it using the given `SECRET` passphrase, and installs a license file for all users of the computer. Manual activation files can be downloaded from the My Products page in the LiveCode account management site.

It is also possible to deactivate LiveCode with:

```
<livecode> deactivate
```

Since LiveCode is actually a GUI application, it needs to be run slightly differently from other command-line programs.

On Windows, the command is:

```
start /wait <livecode> activate -file LICENSE -passphrase SECRET
start /wait <livecode> deactivate
```

On Mac OS X, you need to do:

```
<livecode>/Contents/MacOS/LiveCode activate -file LICENSE -passphrase SECRET
<livecode>/Contents/MacOS/LiveCode deactivate
```

New Features

macOS

Apple architecture support

Native support for the arm64-based Apple architecture has been added to the macOS IDE, standalone, and server engine.

The IDE and server engine are now shipped as universal binaries with both an Intel and Apple architecture slice.

All externals shipped with LiveCode, except for XPDF and the dboracle revdb driver, also now support Apple architecture.

You can determine which architecture a running macOS engine is using with `the processor` function in LiveCode Script, or the `the architecture` syntax in LiveCode Builder. If running as Intel architecture, these will return `x86_64`; and if running as Apple architecture, these will return `arm64`.

Note: Apple architecture support is currently experimental. To run the IDE using Apple architecture (on supported machines), you must toggle the `Open using Rosetta` option in the `LiveCode.app` bundle's `Get Info` pane in Finder.

Note: As the server engine is a bare executable, it will run using the native architecture of the host machine. If this is of concern, either use the `arch` shell command to force it run using Intel architecture, or the `lipo` shell command to remove the Apple architecture slice.

Standalone Builder

Apple architecture support

Support for building macOS standalones which have native Apple architecture support has been added to the standalone builder.

You can now select whether you want to build a macOS standalone with an Intel slice, an Apple slice, or both on the macOS pane of standalone settings.

Note: Apple architecture support is currently experimental. To build a standalone with native Apple architecture support you must explicitly choose the `macOS Apple` option in standalone settings.

Breaking Changes

LiveCode Script

Correction to system version string on macOS

The engine will now return the correct value for `the systemVersion` on all macOS versions.

Previously, when running on macOS 11.0 and above, the system version would be reported as `10.16.0`.

Now, on Big Sur, the system version will be reported starting from `11.0.0`; and, on Monterey, the system version will be reported starting from `12.0.0`.

Due to this change, any legacy (macOS-specific) scripts which assume that the first number of `the systemVersion` will be `10` to detect running on Mac OS X, or ignore the first number of `the systemVersion` entirely and only compare using the second and third, will have to be updated to accomodate this change.

Issues Resolved

Features implemented

22998	Apple architecture support has been added to the macOS IDE, standalone builder, and server engine	9.6.8-rc-1

Regressions fixed

23757	The standalone builder now ensures all executables in a macOS standalone only contain the requested architectures	9.6.8-rc-3
23733	Using `export` when the source is an image object of a closed stack will no longer cause an error to be thrown	9.6.8-rc-2
	LiveCode will no longer crash on startup on versions of macOS El Capitan	9.6.8-

23734	(10.11) and below	rc-2

Bugs fixed

23723	The property inspector geometry pane icons have been reinstated	9.6.8-rc-2
23740	Building a macOS standalone on macOS Sierra (10.12) and earlier will not include an Apple architecture slice, even if selected	9.6.8-rc-2
20546	The `revpdfprinter` driver is now included with the server engine allowing `open printing to pdf` to work	9.6.8-rc-1
22650	Repeatedly setting the filename of a mobile player control on iOS now works reliably	9.6.8-rc-1
22887	On macOS, `the systemVersion` will now return the correct value on all operating system versions	9.6.8-rc-1
23515	The default deployment platforms for a new stack are now appropriate for the current license	9.6.8-rc-1
23527	Using `exit to top` in a function called when evaluating a function call's arguments now works correctly	9.6.8-rc-1
23576	AppleScript can now be used to control other applications from both the IDE and standalones on macOS	9.6.8-rc-1
23578	The XPDF external is now available again	9.6.8-rc-1
23579	The minWidth, maxWidth, minHeight and maxHeight properties now work correctly on stacks which have their scaleFactor set	9.6.8-rc-1
23587	Exiting fullscreen media playback will no longer cause a crash on iOS	9.6.8-rc-1
23602	The (legacy) Android app icon standalone setting will now persist from older versions of LiveCode	9.6.8-rc-1
23611	The return value of `do as applescript` is now formatted correctly when running through Rosetta on Apple architecture macs	9.6.8-rc-1
23612	The map widget API key is now reliably included in the standalone manifest	9.6.8-rc-1
23618	Folders added to the `Copy Files` list are now included recursively in web standalones	9.6.8-rc-1
23620	Adding empty files to the `Copy Files` list will no longer cause standalone building to fail when targetting iOS	9.6.8-rc-1
23621	The supported platforms of the `INI Library`, `Mac Status Menu`, `Secure Key Storage`, `Time zone library` and `Drawing library` extensions are now displayed correctly in the standalone builder	9.6.8-rc-1
23626	Encoding text to `utf16be` using the `textEncode` function now works correctly on Windows	9.6.8-rc-1
23638	The user guide PDF no longer has missing images	9.6.8-rc-1
23655	Using the image variant of `export` when the source is not an image object will no longer cause a crash.	9.6.8-rc-1

23661	Updating the screen in a tight loop on macOS will no longer cause memory usage to slowly increase	9.6.8-rc-1
23663	The dictionary entries for `mergJSON` are now available again	9.6.8-rc-1
23673	The `put` command can now be used without a target before `else` in a single-line `if` statement	9.6.8-rc-1
23682	Redraw glitches no longer occur when performing a visual effect immediately after another window update on macOS	9.6.8-rc-1
23705	Using the widget variant of `export` when the source is an unknown widget kind will no longer cause a crash	9.6.8-rc-1
12550	The time taken to clone a stack when the message box is open has been reduced	9.6.7
23492	IDE responsiveness will no longer reduce after the message box has been opened and then closed	9.6.7
23531	Script execution errors are now displayed correctly when the script editor is opened in debug mode	9.6.7

Previous release notes

- LiveCode 9.6.7 Release Notes
- LiveCode 9.6.6 Release Notes
- LiveCode 9.6.5 Release Notes
- LiveCode 9.6.4 Release Notes
- LiveCode 9.6.3 Release Notes
- LiveCode 9.6.2 Release Notes
- LiveCode 9.6.1 Release Notes
- LiveCode 9.6.0 Release Notes
- LiveCode 9.5.1 Release Notes
- LiveCode 9.5.0 Release Notes
- LiveCode 9.0.5 Release Notes
- LiveCode 9.0.4 Release Notes
- LiveCode 9.0.3 Release Notes
- LiveCode 9.0.2 Release Notes
- LiveCode 9.0.1 Release Notes
- LiveCode 9.0.0 Release Notes
- LiveCode 8.1.10 Release Notes
- LiveCode 8.1.9 Release Notes
- LiveCode 8.1.8 Release Notes
- LiveCode 8.1.7 Release Notes
- LiveCode 8.1.6 Release Notes
- LiveCode 8.1.5 Release Notes
- LiveCode 8.1.4 Release Notes
- LiveCode 8.1.3 Release Notes
- LiveCode 8.1.2 Release Notes
- LiveCode 8.1.1 Release Notes
- LiveCode 8.1.0 Release Notes

- LiveCode 8.0.2 Release Notes
- LiveCode 8.0.1 Release Notes
- LiveCode 8.0.0 Release Notes
- LiveCode 7.1.4 Release Notes
- LiveCode 7.1.3 Release Notes
- LiveCode 7.1.2 Release Notes
- LiveCode 7.1.1 Release Notes
- LiveCode 7.1.0 Release Notes
- LiveCode 7.0.6 Release Notes
- LiveCode 7.0.4 Release Notes
- LiveCode 7.0.3 Release Notes
- LiveCode 7.0.1 Release Notes
- LiveCode 7.0.0 Release Notes
- LiveCode 6.7.11 Release Notes
- LiveCode 6.7.10 Release Notes
- LiveCode 6.7.9 Release Notes
- LiveCode 6.7.8 Release Notes
- LiveCode 6.7.7 Release Notes
- LiveCode 6.7.6 Release Notes
- LiveCode 6.7.4 Release Notes
- LiveCode 6.7.2 Release Notes
- LiveCode 6.7.1 Release Notes
- LiveCode 6.7.0 Release Notes
- LiveCode 6.6.5 Release Notes
- LiveCode 6.6.4 Release Notes
- LiveCode 6.6.3 Release Notes
- LiveCode 6.6.2 Release Notes
- LiveCode 6.6.1 Release Notes
- LiveCode 6.6.0 Release Notes
- LiveCode 6.5.2 Release Notes
- LiveCode 6.5.1 Release Notes
- LiveCode 6.5.0 Release Notes
- LiveCode 6.1.3 Release Notes
- LiveCode 6.1.2 Release Notes
- LiveCode 6.1.1 Release Notes
- LiveCode 6.1.0 Release Notes
- LiveCode 6.0.2 Release Notes
- LiveCode 6.0.1 Release Notes
- LiveCode 6.0.0 Release Notes