LiveCode 9.6.6-rc-2 Release Notes

Overview

This document describes all the changes that have been made for LiveCode 9.6.6-rc-2, 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.

Mac

The Mac engine supports:

- 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)

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
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

LiveCode Script

iOS device safe area insets function

An `iphoneSafeAreaInsets` function has been added to enable getting the safe area insets of the device the app is running on. These can be used to adjust your app's user interface elements to avoid areas that are covered by device furniture, e.g. the notch.

Standalone Builder

Android adaptive icons

It is now possible to configure your Android apps to use adaptive icons.

To use adaptive icons, generate them using the `Image Asset Editor` in `Android Studio` and then choose the resulting `res` folder in the android standalone settings pane.

Android app query whitelist

Support has been added for specifying which other apps can be interacted with via the `intent` and other (custom) URL schemes.

Since Android 11, by default, an app cannot interact with any others due to changes to package visibility.

You must now specify which apps you need access to by adding their ids to the `App ID Query Whitelist` field in the Android standalone settings.

Issues Resolved

Features implemented

22518	It is now possible for Android apps to be built with adaptive icons	9.6.6-rc-1
23439	It is now possible for Android apps to specify which other apps are accessible via URL schemes	9.6.6-rc-1
23463	A new function `iphoneSafeAreaInsets` has been added which returns the safe area insets of the current device	9.6.6-rc-1

Bugs fixed

23496	**The example code in the `iphoneSafeAreaInsets` function dictionary entry has been corrected**	**9.6.6-rc-2**
23500	**Saving a standalone for iOS 15 or later now works correctly**	**9.6.6-rc-2**
8169	Opening HyperCard stacks will no longer cause a crash	9.6.6-

		rc-1
11726	Support for the system date and the system time has been added on Android	9.6.6-rc-1
14344	A stack's rect will no longer become out of sync with its actual position after being moved in the IDE on macOS	9.6.6-rc-1
14436	A wider range of TrueType font files will now work on Android and Web	9.6.6-rc-1
17579	Use of the term `Tutorials` have been replaced by the term `Lessons` throughout the IDE	9.6.6-rc-1
18529	When a runtime error occurs in a context where the script debugger cannot run, the error dialog will be displayed rather than failing silently	9.6.6-rc-1
19419	Conversions between global and local co-ordinates are now correct on multi-screen macOS systems.	9.6.6-rc-1
19937	The welcome tutorial can now be started on first run via the `Start Interative Welcome` button in the start center	9.6.6-rc-1
20045	Starting a new tutorial will now close the previous tutorial's stack, if present	9.6.6-rc-1
21604	The performance of the script editor while typing has been improved	9.6.6-rc-1
21937	Downloading large amounts of data using tsNet on Windows is now substantially faster	9.6.6-rc-1
23088	Invalid PDFs will no longer be generated when using the `open printing to PDF` command on Windows using the 64-bit engine	9.6.6-rc-1
23091	Script debugging will no longer fail to work intermittently when running the IDE with pro features	9.6.6-rc-1
23117	Scripts which run without locking the screen are no longer slower on macOS Big Sur and later	9.6.6-rc-1
23216	The drawing library now correctly handles gradients which are referred to using an id containing `-` or `_`	9.6.6-rc-1
23263	Key events will now work correctly after a modal dialog is shown from a mouseUp handler in a grouped control	9.6.6-rc-1
23271	The `iphoneDeviceModel` function now returns the correct model string when run on an iOS/iPadOS simulator.	9.6.6-rc-1
23279	Default buttons and progress bars will no longer cause unnecessary CPU usage on macOS 10.10 (Yosemite) or later	9.6.6-rc-1
23294	The `byteOffset` function will no longer cause incorrect results in some cases nor cause a crash	9.6.6-rc-1
23295	Scripts containing accented characters will no longer incorrectly report being externally modified	9.6.6-rc-1
23299	Using the `print link` command when printing to PDF on Windows no longer causes a crash	9.6.6-rc-1
23305	The copyright notices in the engine app bundle (macOS) and exe (Windows) have been updated	9.6.6-rc-1
23336	A significant memory leak in the browser widget on macOS has been resolved	9.6.6-rc-1
	POST data larger than 64kb is now parsed correctly by the Windows server	9.6.6-

23340	engine	rc-1
23366	Audio-only players no longer consume excessive CPU when in Edit Mode on macOS Big Sur	9.6.6-rc-1
23394	WebGL content now displays in the browser widget when running on macOS 12.x (Monterey)	9.6.6-rc-1
23415	The `union` (and other set operation) commands now throw an error if there is no `into` clause and the target is not a declared variable	9.6.6-rc-1
23418	Building iOS apps using Xcode 13.2 with the iOS 15.2 SDK is now supported	9.6.6-rc-1
23437	Local file and content access has been re-enabled in the browser widget and native control on Android	9.6.6-rc-1
23459	Touch events now occur at the correct location when handled by objects underneath a disabled Android scroller	9.6.6-rc-1
23466	SVG elements with a `stroke-width` of 0 are no longer drawn with a hairline stroke	9.6.6-rc-1
23474	There is no longer a delay when opening the IDE menus on macOS	9.6.6-rc-1
23475	Selecting a specific browser to test a web project against now works correctly	9.6.6-rc-1
23486	The overhead of using `do` to evaluate long `JavaScript` scripts in the browser widget on Android has been greatly reduced	9.6.6-rc-1

Previous 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