LiveCode 9.6.2-rc-5 Release Notes

Overview

LiveCode 9.0 enables access to libraries and platform APIs written in many other languages thanks to the community-funded 'Infinite LiveCode' project.

This includes a greatly improved LiveCode Builder virtual machine.

LiveCode 9.0 contains many additional improvements to support LiveCode app developers, including:

- A new "spinner" widget
- OAuth2 authentication library for use with web APIs (e.g. Facebook, Google and GitHub)
- A command argument parser library for building command-line standalones
- Updates and performance improvements for existing widgets

Known issues

- The installer will currently fail if you run it from a network share on Windows. Please copy the installer to a local disk before launching on this platform.
- The browser widget does not work on 32-bit Linux.
- 64-bit standalones for Mac OS X do not have support for audio recording.

Breaking changes

Boolean constants

In this release, boolean constants `true` and `false` have been changed so that they resolve to values of boolean type (rather than string). This will affect any uses of the `is strictly` operator on such values, i.e. previously the following were true:

true is strictly a string false is strictly a string

Now, they are both false, and the following are true:

true is strictly a boolean false is strictly a boolean

Boolean constants passed as elements of arrays to LCB handlers will not require conversion to boolean values in LCB - in fact any attempt to do so assuming they are strings will cause an error. Any array elements which are intended to be booleans in LCB should be checked for their type before conversion. For example, any of the following could be done by an LCB library user:

```
put true into tArray["enabled"]
put "true" into tArray["enabled"]
put (tVar is not "enabled") into tArray["enabled"]
```

An LCB handler to which `tArray` is passed should do the following:

```
variable tEnabled as Boolean
if tArray["enabled"] is a boolean then
    put tAction["enabled"] into tEnabled
else
    put tAction["enabled"] parsed as boolean into tEnabled
end if
```

Infinity constant

The constant `infinity` has been added to the language in this release. As a result, the unquoted literal `infinity` is now reserved. Any existing uses of it should be quoted, as otherwise it will resolve to the floating point value representing infinity, rather than the string "infinity".

Implicit object

A number of LCB commands use an implicit object to provide context for their execution. Some of these commands also allow specifying an explicit object. These commands are:

- `execute script`
- `send`
- `post`
- `image from file`
- `resolve file` - new in this version

In previous releases `execute script` and `image from file` would use `this card of the defaultStack` as the implicit object even if called from a widget. The `send` and `post` commands, however, used `this card of the defaultStack` when in a library module handler and the host widget when in a widget module handler. This release changes `execute script` and `image from file` to also use the host widget as the implicit object. This means, for example, that `image from file` will resolve a relative file path relative to the `stackFile` the host widget is on rather than the `stackFile` of the `defaultStack`.

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

Note: On 64-bit Windows installations, LiveCode runs as a 32-bit application through the WoW layer.

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.

Note: It may be possible to compile and run LiveCode Community for Linux on other architectures but this is not officially supported.

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)

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 8.2 on MacOS X 10.11
- Xcode 9.2 on MacOS 10.12 (Note: You need to upgrade to 10.12.6)
- 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 12.1 on MacOS 10.15 (Note: You need to upgrade to 10.15.4)

It is also possible to set other versions of Xcode, to allow testing on a wider range of iOS simulators. For instance, on MacOS 10.12 (Sierra), you can add *Xcode 8.2* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 10.2*.

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

- 10.2 (included in Xcode 8.2)
- 11.2 (included in Xcode 9.2)
- 12.1 (included in Xcode 10.1)
- 13.2 (included in Xcode 11.3)
- 14.1 (included in Xcode 12.1)

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)

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 Indy or Business edition

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 for LiveCode Indy or Business edition

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
```

LiveCode Community engine changes

Specific engine bug fixes (9.6.2-rc-5)

23154 **Fix engine lockup when restoring window after maximizing on Macos Big Sur**

Specific engine bug fixes (9.6.2-rc-4)

23119 Fix script editor unresponsiveness while scrolling on Macos Big Sur

Specific engine bug fixes (9.6.2-rc-3)

23075 Fix 'ask ... as sheet' dialog unresponsive when launched from a modal stack
23079 Include missing mergLA builds
23085 Fix visual effect rendering on Macos Big Sur

Specific engine bug fixes (9.6.2-rc-2)

22997 Updated list of supported Apple platforms.
23003 Fix long pause after closing modal dialog on MacOS X
23019 Fix window redraw issue on MacOS X when making changes to a card immediately after switching to it
23053 Fix the color dialog failing to respond to input events when launched from a modal stack on macOS

Specific engine bug fixes (9.6.2-rc-1)

14941 Corrected XML function references in the User Guide
19016 Ensure printing a field with vGrid enabled does not print black rectangles on Windows
21471 Fix revIsSpeaking returning incorrect values on Windows
22143 Fix interactive tutorial on Windows
22586 Fix crash in `split` command with multi-char delimiter
22877 Restore mouse focus correctly after relayer command
22879 Fix incorrect text color used for tab buttons on MacOS Big Sur
22880 Fix windowShape not being applied to stacks on MacOS Big Sur
22888 Add support for building apps against iOS 14.1 SDK
22891 The behavior of the rename command when the target file exists has been clarified.
22892 Corrected the definition of is a number and integer
22894 Ensure children nodes of the tag can be merged when additional android manifests are added
22896 Fix buttons on answer dialog opened as sheet not responding on MacOS Big Sur
22918 Fix crash when deleting an object while handling a message sent to that object
22927 Fix drag-and-drop not working within a modal stack
22929 Fix excessive CPU usage while showing modal dialog
22933 Fix excess time taken to begin new canvas layer
22935 Fix engine instability after opening & closing a modal stack
22940 Remove support for 32bit on MacOS IDE and standalones
22942 Ensure the shellCommand defaults to COMSPEC on Windows
22947 The long deprecated `iphoneSystemIdentifier` command now returns empty
22951 Added note about scaleFactor not persisting in standalones.

22952	Fix modal stack not responding to input events if opened in the "openStack" handler of another modal stack
22959	Corrected missing text issue in time dictionary enty.
22963	Fix incorrect result when split does not find multi-char delimiter
22974	Fix crash in mobilePickPhoto on Android 11
22980	Menu items which don't have a mark now display correctly on Big Sur when running on ARM processors

LiveCode Community IDE changes

Specific IDE bug fixes (9.6.2-rc-1)

15638	Remove `cantModify` from stack basic properties palette
22847	Improve display of enum values in dictionary
22873	Added missing text to the entry for dgEditMode.
22897	Ensure colorization in the script editor is preserved on MacOS Big Sur with Source Code Pro fonts.
22903	Ensure RowLeftSwipeControlClicked message is sent with a target parameter

Previous 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 8.1.9 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 8.1.10 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 6.7.9 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 6.7.11 Release Notes
- LiveCode 6.7.10 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