LiveCode 9.0.5-rc-1 Release Notes

- Overview
- Known issues
- Breaking changes
  - Standalone Building
  - LiveCode Builder
- Platform support
  - Windows
  - Linux
  - Mac
  - iOS
  - Android
  - HTML5
- Setup
  - Installation
  - Uninstallation
  - Reporting installer issues
  - Activating LiveCode Indy or Business edition
  - Command-line installation
  - Command-line uninstallation
  - Command-line activation for LiveCode Indy or Business edition
- LiveCode Community engine changes
  - Specific engine bug fixes (9.0.5-rc-1)
- LiveCode Community IDE changes
  - Specific IDE bug fixes (9.0.5-rc-1)
- LiveCode Indy engine changes
  - Specific engine bug fixes (9.0.5-rc-1)
- LiveCode builder changes
  - Specific builder bug fixes (9.0.5-rc-1)
- Previous 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**

**Standalone Building**

The standalone builder has always needed to close the stacks it builds for reasons pretty deeply ingrained in the code. However this causes a few problems, for example:

- values in script locals become empty
- behaviors are broken when the parent script is on / in a stack which closes

As an attempt to improve this situation, the code that locks messages when closing and opening stacks for standalone builds has been removed. This means that where previously mainstacks would not receive any of the (pre)open* and close* messages (e.g. preOpenStack, openStack, openCard, closeStack etc) during standalone build, they now do.

If this causes problems for your stack, you can exit from the handler if standalone building is in progress:
LiveCode Builder

Exponentiation operator precedence

Prior to this release, exponentiation had lower precedence that unary minus. In order to write code that operates as expected in both this release and previous releases, please use parentheses where appropriate.

Using lc-compile tool in LiveCode 9:

\[-1^2 = -1\]

Using lc-compile tool in LiveCode 8:

\[-1^2 = 1\]

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)

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 6.2 on MacOS X 10.9
- Xcode 6.2 and 7.2 on MacOS X 10.10
- 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)

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 6.2 in the Mobile Support preferences, to let you test your stack on the iOS Simulator 8.2.

We currently support deployment for the following versions of iOS:

- 8.2 [simulator]
- 9.2
- 10.2
- 11.2
- 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 ARMv7 or ARMv8 processors. It will run on the following versions of Android:

- 4.1-4.3 (Jelly Bean)
- 4.4 (KitKat)
- 5.0-5.1 (Lollipop)
- 6.0 (Marshmallow)
- 7.x (Nougat)
- 8.x (Oreo)

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:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>&lt;x86 program files folder&gt;/RunRev/LiveCode &lt;version&gt;</td>
</tr>
<tr>
<td>Linux</td>
<td>/opt/livecode/livecode-&lt;version&gt;</td>
</tr>
</tbody>
</table>

The installations when installing for “This User” are:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>&lt;user roaming app data folder&gt;/RunRev/Components/LiveCode &lt;version&gt;</td>
</tr>
<tr>
<td>Linux</td>
<td>~/.runrev/components/livecode-&lt;version&gt;</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2000/XP</td>
<td>&lt;documents and settings folder&gt;/&lt;user&gt;/Local Settings/</td>
</tr>
<tr>
<td>Windows Vista/7</td>
<td>&lt;users folder&gt;/&lt;user&gt;/AppData/Local/RunRev/Logs</td>
</tr>
<tr>
<td>Linux</td>
<td>&lt;home&gt;/.runrev/logs</td>
</tr>
</tbody>
</table>

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 OPTIONS:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Install the IDE for &quot;All Users&quot;. If not specified, LiveCode will be installed</td>
</tr>
</tbody>
</table>
-all
Option for the current user only.

-deskshortcut
Place a shortcut on the Desktop (Windows-only)

-startmenu
Place shortcuts in the Start Menu (Windows-only)

-location
The folder to install into. If not specified, the LOCATION defaults to those described in the "Installation" section above.

-log
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 (9.0.5-rc-1)

- **12419** Fix touch messages not being sent to controls under an mobile scroller control on Android
- **16947** Fix memory leak when calling self-recursive handlers having reference parameters
- **17315** Update the `delete` variable entry to clarify only keys are deleted
- **20022** Updated an external link to Apple Event documentation in dictionary entries regarding Apple Events.
- **20027** Fix value assignment to undeclared variable in `matchText`
- **21156** Deprecate `videoClip` object in documentation
- **21916** Fix memory leak when reconfiguring the printer on macOS
- **21928** Fix memory leak when using menu buttons
- **21936** Added to documentation on brush and pencil about how paintings made without existing images will not be kept.
21941  Fix local-ref overflow when repeatedly calling Java from LCB on older Android OSes
21945  Fix touch events being sent to and controlAtLoc returning hidden and disabled widgets
21955  Eliminate 'Invalid TIFF' (console) warning on startup on macOS
21956  Fix memory leak in repeat for each chunk when the chunk is of string type
21958  Fix non-effective buffer overrun on initial seeding of pseudo-random number generator
21959  Fix memory leak when using clipboard functionality
21960  Ensure private handler lookups are always cached
21986  Fix benign leak of backdrop window on engine shutdown on macOS
21987  Fix pasteboard leak on macOS
21988  Fix memory leaks when using clipboard functions
21989  Fix benign leak of apple events on startup on macOS
21990  Fix memory leak when destroying a stack's window on macOS
21991  Ensure external library code modules are unloaded when no longer referenced
21992  Fix memory leaks when sorting cards, fields, or object selection
21993  Make name and string values more memory efficient
21994  Fix memory leak when using arrayDecode on 7.0 format encoded arrays
21995  Fix memory leaks when using sockets
21996  Fix memory leak when using filter elements of array
21997  Fix memory leaks when splitting strings in specific cases
21998  Use name values rather than string values for variable initializers
21999  Fix memory leak when converting strings to numbers
22000  Fix memory leak when parsing save with format
22001  Fix memory leak when using import eps command
22002  Fix memory leak when setting numberFormat properties
22003  Fix memory leaks occurring through the use of autocomplete in the IDE
22004  Fix memory leak when configuring syntax highlighting in the script editor
22005  Fix memory leaks from using the browser widget
22006  Fix memory leaks when using player on macOS
22007  Fix memory leaks when using menus on macOS
22008  Fix memory leak when parsing menu strings
22009  Fix memory leaks when releasing loaded LCB modules and instances
22010  Fix memory leak when binding some LCB foreign handlers
22011  Fix memory leak when using repeat for each in LCB in early termination cases
22012  Fix memory leak when using LCB's log command on desktop platforms
22013  Fix memory leak when using send/post/execute script from LCB
22015  Fix memory leak when bridging foreign values to LCS
22016  Fix memory leak when setting widget properties in some cases
22017  Fix memory leak when updating listBehavior fields in some cases
22019  Fix several memory leaks occurring in use of specific field features
22020 Fix memory leak when dbsqlite connection fails
22024 Fix LCB’s ‘is among the elements of’ when searching for array elements
22042 Updated SQLite to version 3.28.0

LiveCode Community IDE changes

Specific IDE bug fixes (9.0.5-rc-1)

20701 Make manual height of PI consistent, increase value field in Custom Props
21414 Fix error when closing the Script Editor and the Find window is still open
21484 Allow scroll wheel to work in PI text fields
21834 Fix display of unresolvable behaviors and flag them in PB
21896 Fix Autocomplete IF-structs
21969 Ensure the templategroup is reset before creating the tools palette
22014 Ensure no dialog is shown about closing the last stack in file when building a standalone
22033 Keep value of global props linkcolor and underlinelink
22036 Allow setting the label of a graphic from the Property Inspector

LiveCode Indy engine changes

Specific engine bug fixes (9.0.5-rc-1)

21948 Fix crash when setting the zoomFactor of a camera control on iOS

LiveCode builder changes

Specific builder bug fixes (9.0.5-rc-1)

22047 Allow nothing to be passed as a variadic argument to a foreign handler
22060 Fix bug in canvas library that can result in Points being initialized with previously used values.
22063 Ensure nothing can be passed to an optional foreign handler type parameter
Previous 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