LiveCode 6.6.0-rc-1 Release Notes

Table of contents

Overview

This document describes all the changes that have been made for LiveCode 6.6.0-rc-1, including bug fixes and new syntax.

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.

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

The engine supports the following Windows OSes:

- Windows XP SP2 and above
- Windows Server 2003
- Windows Vista SP1 and above (both 32-bit and 64-bit)
- Windows 7 (both 32-bit and 64-bit)
- Windows Server 2008
- Windows 8.x (Desktop)

Note: *On 64-bit platforms the engine still runs as a 32-bit application through the WoW layer.*

Linux

The linux engine requires the following:

- 32-bit installation, or a 64-bit linux distribution that has a 32-bit compatibility layer
- 2.4.x or later kernel
- X11R5 capable Xserver running locally on a 24-bit display
- glibc 2.3.2 or later
- gtk/gdk/glib (optional – required for native theme support)
- pango/xft (optional – required for pdf printing, anti-aliased text and unicode font support)
- lcms (optional – required for color profile support in JPEGs and PNGs)
- gksu (optional – required for elevate process support)

Note: *The optional requirements (except for gksu and lcms) are also required by Firefox and Chrome, so if your linux distribution runs one of those, it will run the engine.*
Note: *If the optional requirements are not present then the engine will still run but the specified features will be disabled.*
Note: *LiveCode and standalones it builds may work on remote Xservers and in other bit-depths, however this mode of operation is not currently supported.*

Mac

The Mac engine supports:

- 10.5.8 and later (Leopard) on Intel and PowerPC
- 10.6.x (Snow Leopard) on Intel
- 10.7.x (Lion) on Intel
- 10.8.x (Mountain Lion) on Intel
- 10.9.x (Mavericks) on Intel

Note: *The engine runs as a 32-bit application regardless of the capabilities of the underlying processor.*

Setup

Installation

Each distinct version has its own complete folder – multiple versions will no longer install side-by-side: on Windows (and Linux), each distinct version will gain its own start menu (application menu) entry; on Mac, each distinct version will have its own app bundle.
The default location for the install on the different platforms when installing for 'all users' are:

- Windows: <x86 program files folder>/RunRev/ LiveCode 6.6.0-rc-1
- Linux: /opt/runrev/livecode-6.6.0-rc-1
- Mac: /Applications/ LiveCode 6.6.0-rc-1.app

The default location for the install on the different platforms when installing for 'this user' are:

- Windows: <user roaming app data folder>/RunRev/Components/LiveCode 6.6.0-rc-1
- Linux: ~/.runrev/components/livecode-6.6.0-rc-1
- Mac: ~/Applications/ LiveCode 6.6.0-rc-1.app

Note: *If your linux distribution does not have the necessary support for authentication (gksu) then the installer will run without admin privileges so you will have to manually run it from an admin account to install into a privileged location.*

Uninstallation

On Windows, the installer hooks into the standard Windows uninstall mechanism. This is accessible from the appropriate pane in the control panel.
On Mac, simply drag the app bundle to the Trash.
On Linux, the situation is currently less than ideal:

- open a terminal
- *cd* to the folder containing your rev install. e.g.

```
cd /opt/runrev/livecode-6.6.0-rc-1
```

- execute the *.setup.x86* file. i.e.

```
./.setup.x86
```

- follow the on-screen instructions.

Reporting installer issues

If you find that the installer fails to work for you then please file a bug report in the RQCC or email support@runrev.com so we can look into the problem.
In the case of failed install it is vitally important that you include the following information:

- Your platform and operating system version
- The location of your home/user folder
- The type of user account you are using (guest, restricted, admin etc.)
- The installer log file located as follows:
- **Windows 2000/XP:** <documents and settings folder>/<user>/Local Settings/

- **Windows Vista/7:** <users folder>/<user>/AppData/Local/RunRev/Logs
- **Linux:** <home>/.runrev/logs
- **Mac:** <home>/Library/Application Support/Logs/RunRev

Activation

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.

Multi-user and network install support (4.5.3)

In order to better support institutions needing to both deploy the IDE to many machines and to license them for all users on a given machine, a number of facilities have been added which are accessible by using the command-line.
Note: *These features are intended for use by IT administrators for the purposes of deploying LiveCode in multi-user situations. They are not supported for general use.*

Command-line installation

It is possible to invoke the installer from the command-line on both Mac and Windows. When invoked in this fashion, no GUI will be displayed, configuration being supplied by arguments passed to the installer.
On both platforms, the command is of the following form:

<exe> install noui *options*

Here *options* is optional and consists of one or more of the following:

-allusers	Install the IDE for all users. If not specified, the install will be done 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 location to install into. If not specified, the location defaults to those described in the *Layout* section above.
-log *logfile*	A file to place a log of all actions in. If not specified, no log is generated.

Note that the command-line variant of the installer does not do any authentication. Thus, if you wish to install to an admin-only location you will need to be running as administrator before executing the command.
As the installer is actually a GUI application, it needs to be run slightly differently from other command-line programs.
In what follows <installerexe> should be replaced with the path of the installer executable or app (inside the DMG) that has been downloaded.
On Windows, you need to do:

start /wait <installerexe> install noui *options*

On Mac, you need to do:

"<installerexe>/Contents/MacOS/installer" install noui *options*

On both platforms, the result of the installation will be written to the console.

Command-line activation

In a similar vein to installation, it is possible to activate an installation of LiveCode for all-users of that machine by using the command-line. When invoked in this fashion, no GUI will be displayed, activation being controlled by any arguments passed.
On both platforms, the command is of the form:

<exe> activate -file *license* -passphrase *phrase*

This command will load the manual activation file from *license*, decrypt it using the given *passphrase* and then install a license file for all users of the computer. Manual activation files can be downloaded from the 'My Products' section of the RunRev customer accounts area.
This action can be undone using the following command:

<exe> deactivate

Again, as the LiveCode executable is actually a GUI application it needs to be run slightly differently from other command-line programs.
In what follows <livecodeexe> should be replaced with the path to the installed LiveCode executable or app that has been previously installed.
On Windows, you need to do:

start /wait <livecodeexe> activate -file *license* -passphrase *phrase*
start /wait <livecodeexe> deactivate

On Mac, you need to do:

"<livecodeexe>/Contents/MacOS/LiveCode" activate -file *license* -passphrase *phrase*
"<livecodeexe>/Contents/MacOS/LiveCode" deactivate

On both platforms, the result of the activation will be written to the console.

Proposed changes

The following changes are likely to occur in the next or subsequent non-maintenance release:

- The engine (both IDE and standalone) **will require** gtk, gdk, glib, pango and xft on Linux

Engine changes

'assert' command (6.6.0-rc-1 - experimental)

An experimental command has been added to support writing tests. The assert command has two forms.

The first form evaluates an expression and checks to see if the it evaluates to true or false:

assert *expr*
assert true *expr*
assert false *expr*

The second form evaluates an expression and checks to see if it throws an error or not:

assert success *expr*
assert failure *expr*

In either case if the condition fails, then an **assertError** message is sent to the object containing the command:

assertError *handlerName*, *line*, *column*, *objectLongId*

Examples:

assert 1 is 1 -- succeeds
assert true "black" is "white" -- fails
assert false "hello" is a number -- succeeds
assert false 1 + 1 = 2 -- fails

assert success 1 + 1 -- succeeds
assert failure 1 + "z" -- fails

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

New showAll fullscreenmode. (6.6.0-rc-1)

This adds a new option for the stack **fullscreenmode** property:

- "showAll" - scale the stack preserving aspect ratio so all content within the stack rect is visible. Portions of the stack outside the stack rect will be visible if the scaled stack does not fit the screen exactly.

Hi-DPI support for Windows 7/8 and OSX. (6.6.0-rc-1)

Hi-DPI support has been added for Windows 7/8 and OSX.

The global **pixelScale** property is no longer settable on desktop environments.

Desktop systems may have multiple displays attached, each with their own density value. On platforms that support different scale values for each display, LiveCode will now automatically render each stack at the correct scale for the screen on which it is displayed.

A new global property has been added to allow this feature to be turned on or off if required:

- The **usePixelScaling** property controls whether or not LiveCode will automatically apply pixel scaling. If set to false, pixel scaling will be disabled and LiveCode will draw at a 1:1 scale, allowing the operating system to perform any scaling required.

When **usePixelScaling** is false, the **pixelScale** will be set to 1 and will not be modifiable.

Due to limitations of the platform, applications on Windows will not be able to enable or disable pixel scaling. On Android there is no equivalent OS-level scaling provided, so pixel scaling will always be used.

Handling multiple displays

As there may be multiple displays with differing scale values, The global **systemPixelScale** property will now return the maximum screen density for all connected displays.

Two additional global properties have been added to provide the pixel scale values when multiple displays are available:

- The **screenPixelScale** returns the pixel scale of the main screen
- The **screenPixelScales** returns a return-delimited list of the pixel scale of each connected display

HTTPS through proxy (6.6.0-rc-1)

The desktop version of LibURL has been updated to support fetching HTTPS URLs through a proxy server.

Image Filtering Updates (6.6.0-rc-1)

Due to a degrade in the image resizing quality in 6.5, the image filtering algorithms have been updated for 6.6.

For LiveCode versions prior to 6.5, the image filtering algorithms were not universal across all platforms. "Good" resize quality used bilinear filtering and "best" used bicubic filtering for all platforms. However, "normal" differed. On Mac, box filtering was used. All other platforms applied no filtering.

For LiveCode versions prior to 6.5, all resize operations were cached (i.e. moving a resized image around did not cause the resize to be recalculated).

For LiveCode 6.5, the image filtering was united across all platforms, with no filtering being applied in "normal" mode, bilinear filtering being used in "good" mode and bicubic filtering being used in "best" mode.

For LiveCode 6.5, only "best" resize operations were cached (the acceleratedRendering mode should be used for cacheing in other modes). All others were calculated on the fly.

The bilinear filter used in 6.5 was of poorer quality when compared to pre 6.5. Additionally, the "normal" mode on Mac was poorer (due to the loss of the box filter). We've addressed this in LiveCode 6.6 by improving the image filtering algorithms across all platforms. "Normal" and "good" mode now use updated algorithms. "Best" mode remains as before.

It should be noted that the improvements to the filters used may cause a slight drop in performance. The final image filters and resize modes used has not been finalized and is open to user input.

Graphics Library Update (6.6.0-rc-1)

The version of the underlying graphics library (LibSkia) used by LiveCode has been updated. This is to allow for new features and performance improvements. The end user should see no significant changes.

OpenSSL & Encryption Updates (6.6.0-rc-1)

Open SSL & encryption support has been added to the iOS and Android engines (using LibOpenSSL version 1.0.1e). This allows developers to use the **encrypt** and **decrypt** commands on the mobile platforms in exactly the same way they would on desktop. In order to include the SSL and encryption libraries, developers must tick the "SSL & Encryption" checkbox in the iOS/Android pane of the standalone builder.

In addition to mobile support, LiveCode now includes its own version of the encryption and SSL libraries on OS X (LibOpenSSL version 1.0.1e). This means developers are now no longer relying on the system installed security libraries on OS X.

Windows and Linux remain unchanged.

Proxy automatic configuration support (6.6.0-rc-1)

LibURL has been updated to add support for proxy auto configuration files (PAC). If no global proxy server has been set using the **HTTPProxy** property, when fetching a URL, LibURL will attempt to parse the systems .pac file in order to extract the proxy server (if any) to use for the given URL.

'secure socket' command (6.6.0-rc-1)

A new command, **secure socket**, has been added. Use **secure socket** to convert an unsecured socket created using *open socket* into a secured socket. This way, all future communications over the socket will be encrypted using SSL.

The secure socket command has 3 variants:

secure socket *socket*
secure socket *socket* **with verification**
secure socket *socket* **without verification**

If 'with verification' is specified, when connecting to a remote peer, the client verifies the peers certificate during the handshake process. The **sslCertificates** can be used to specify a list of certificates to verify against. In addition you can place system wide certificates in System/Library/OpenSSL/certs.

If 'without verification' is specified then peers credentials are not authenticated, and any connection is accepted.

Once secured:

- All pending and future reads from the socket will be assumed to be encrypted.
- All pending writes will complete unencrypted. All future writes will be encrypted.

If the socket fails secure, a **socketError** message is sent to the object that opened the socket (not the object that attempted to secure it).

'#!' now recognised by server (6.6.0-rc-1)

When passing a file to the server engine on the command line, if the first line starts with '#!' it will be treated as a plain livecode script file. This means that '<?' type tags are not treated as blocks of code.

For example the following two are equivalent:

```
#! /usr/bin/livecode-server
put "Hello World!"
```

and

```
<?lc
put "Hello World!"
?>
```

In the former case, the whole file is treated as script. In the latter case, only text within the '<?'/'?>' sections are.

SQLite support updated and improved (6.6.0-rc-1)

The version of SQLite has been updated to 3.8.3.

The SQLite library has been compiled with options - FTS3, FTS4, FTS3_PARANTHESIS and RTREE.

SQLite loadable extensions are now supported. To utilize loadable extensions, the 'extensions' option must be passed to the revOpenDatabase() call when creating the database connection (see below).

Binary data can now be placed into SQLite databases verbatim (without the encoding that used to occur) - this means databases can be made to contain binary data which is compatible with other applications. To utilize this functionality, the 'binary' option must be passed to the revOpenDatabase() call when creating the database connection (see below).

The SQLite revOpenDatabase() call no longer requires 5 arguments and only requires a minimum of 2. It has been updated as follows:

revOpenDatabase("sqlite", *database-file*, [*options*])

The *database-file* parameter is the filename of the SQLite database to connect to.

The *options* parameter is optional and if present should be a comma-separated list of option keywords. The *binary* keyword means place binary data into the database verbatim (without LiveCode encoding). The *extensions* keyword means enable loadable extensions for the connection. Note that the order of the items in the options parameter is not important.

For example:

```
put revOpenDatabase("sqlite", "mydb.sqlite") -- open with legacy binary mode and loadable extensions disabled
put revOpenDatabase("sqlite", "mydb.sqlite", "binary") -- open the connection in the 'new' binary mode
put revOpenDatabase("sqlite", "mydb.sqlite", "extensions") -- enable loadable extensions for this connection
put revOpenDatabase("sqlite", "mydb.sqlite", "binary,extensions") -- enable both 'new' binary mode and loadable extensions
```

Stack scaling (6.6.0-rc-1)

The new **scaleFactor** stack property allows you to set a custom scale factor for a stack.
The **scaleFactor** property accepts a non-zero real number value which represents the scale multiplier.

This can be used when developing stacks that are larger than the available screen space - for example developing a stack to be used on an iPad with a retina display.
You can also preview the appearance of your stack on displays with differing display densities.

Examples:

Scaling a stack to half size:
set the scaleFactor of stack "myLargeStack" to 0.5

Preview the stack appearance on a high-density display:
set the scaleFactor of stack "myApp" to 1.5

OS 10.4 (Tiger) Support (6.6.0-rc-1)

As of version 6.6-dp-1, OS 10.4 (Tiger) support has been dropped from LiveCode. This is primarily for technical reasons: In order to support the latest OS X features (e.g. Cocoa) as well as include the newest versions of thirdparty libraries (e.g. LibSkia, LibSQLite), dropping 10.4 support was required.

Users wishing to produce 10.4 compatible executables can still do so using LiveCode version 6.5.x (and earlier).

Cannot take screen snapshot at high resolution. (6.6.0-rc-1)

The import and export snapshot of screen commands have been updated to allow an 'at size' clause along the same lines as import and export snapshot of object.

If the 'at size' clause is specified, the snapshot of the screen will be taken at device pixel size and then scaled to match the size specified in the at size clause.

For example, to take a snapshot of the screen on a retina display at retina resolution you can use:

import snapshot from rect 0, 0, 400, 400 at size 800, 800

This will take a snapshot of 400x400 logical pixels, which is 800x800 device pixels and then scale to 800x800 pixels - i.e. the result is you get the all the pixels that have been rendered to the retina display.

<> operator is different from 'is not' operator for arrays (6.6.0-rc-1)

The <> operator will now work identically to 'is not' when comparing with an array. Previously, the <> operator would convert arrays to the empty string before comparing.

More specifically:
put 100 into tLeft[1]
put tLeft is empty -- false
put tLeft = empty -- false (same as is)
put tLeft is not empty -- true
put tLeft <> empty -- true (previously this was false)

Note: This is a subtle change which could impact existing scripts. However, since '<>' and 'is not' should be synonyms it is considered that it is more likely that it will fix unnoticed bugs in existing scripts rather than cause them.

acceleratedRendering will clip the end of long scrolling groups (6.6.0-rc-1)

Failing to set image data to the image data of self (6.6.0-rc-1)

SQLite binary entries are non-standard. (6.6.0-rc-1)

The revdb SQLite driver has always encoded any binary data that has been inserted into a column in a non-standard way (the reason for this dates back to V2 of sqlite which couldn't cope with NUL's in data). The

problem with this is that it means binary columns in sqlite databases produced and accessed via LiveCode are incompatible with third-party tools and applications that use SQLite databases.

To resolve this issue, SQLite databases can now be opened in a new mode which turns off the binary encoding and decoding that the driver previously performed.

To open an SQLite database with compatibility treatment of binary data, nothing has changed.

To open an SQLite database with the new binary behavior, pass "binary" as the third parameter of the revOpenDatabase call.

For example:

get revOpenDatabase("sqlite", "mydb.sqlite") -- opens in compatibility binary mode (LiveCode compatible only)
get revOpenDatabase("sqlite", "mydb.sqlite", "binary") -- opens in new binary mode (third-party compatible)

How the driver treats binary data is a per-connection property so different connections to the same database can use different modes if needed.

Specific bug fixes (6.6.0-rc-1)

(bug fixes specific to the current build are highlighted in bold, reverted bug fixes are stricken through)

11874	**BinaryDecode wrong on Android**
11858	**Hidden stacks don't update their position when re-shown**
11844	**Stack height limited to screen height**
11841	**The effective revAvailableHandlers of an object sometimes contains duplicates**
11836	**MouseLoc returns wrong value if pixelScale set to value other than 1**
11816	**Native Android control rects not scaled when set from preOpenStack handler**
11814	**Accelerated rendering causes cards to be rendered in the bottom left of the screen on iOS Retina devices**
11811	**Cannot take screen snapshot at high resolution.**
11789	**mobileComposeMail only blocks the first time on Android.**
11785	**Option menu with height > 22 draws incorrectly on OSX with Retina display.**
11783	**Setting the fullscreenmode on Windows when not fullscreen causes text to change.**
11781	**Visual effects can display in the wrong place on iOS on Retina devices.**
11778	**averageDeviation returns incorrect result on Mac**
11754	**Error (invalid bundle) on uploading app to iOS App Store**
11753	**Cannot play some video streams on Android**
11751	**After selecting an item in an option menu contaning unicode**
11732	**<> operator is different from 'is not' operator for arrays**
11721	**Crash when taking a snapshot of the template graphic**
11720	**SQLite FTS feature doesn't work on iOS or Mac.**
11715	**LiveCode crashes if dragsource object is closed during dragEnd handler**
11703	**iPhoneSetRemoteControlDisplay crashes**
11617	**acceleratedRendering will clip the end of long scrolling groups**
11462	**Failing to set image data to the image data of self**
11442	**Can't quit LiveCode after setting the securityPermissions to "network"**
11124	**No error message when external not found when deploying to simulator**
11069	**mobileComposeMail attachment missing in Android**
10910	**"Crop image" command crashes LiveCode application**

10467 **Indenting of scripts can go wrong**
10280 **SQLite binary entries are non-standard.**
8044 **setting a cprop with quotes loses data**
5331 **Mac live window resizing is off by default.**

Dictionary additions

- **secure socket** (*command*) has been added to the dictionary.
- **scaleFactor** (*property*) has been added to the dictionary.
- **screenPixelScale** (*property*) has been added to the dictionary.
- **screenPixelScales** (*property*) has been added to the dictionary.
- **usePixelScaling** (*property*) has been added to the dictionary.

Dictionary changes

- The entry for **convert** (*command*) has been updated.
- The entry for **decrypt using rsa** (*command*) has been updated.
- The entry for **decrypt** (*command*) has been updated.
- The entry for **encrypt using rsa** (*command*) has been updated.
- The entry for **encrypt** (*command*) has been updated.
- The entry for **find** (*command*) has been updated.
- The entry for **cipherNames** (*function*) has been updated.
- The entry for **revOpenDatabase** (*function*) has been updated.
- The entry for **fullscreenmode** (*property*) has been updated.
- The entry for **pixelScale** (*property*) has been updated.

Previous Release Notes

6.5.2 Release Notes	http://downloads.livecode.com/livecode/6_5_2/LiveCodeNotes-6_5_2.pdf
6.5.1 Release Notes	http://downloads.livecode.com/livecode/6_5_1/LiveCodeNotes-6_5_1.pdf
6.5.0 Release Notes	http://downloads.livecode.com/livecode/6_5_0/LiveCodeNotes-6_5_0.pdf
6.1.3 Release Notes	http://downloads.livecode.com/livecode/6_1_3/LiveCodeNotes-6_1_3.pdf
6.1.2 Release Notes	http://downloads.livecode.com/livecode/6_1_2/LiveCodeNotes-6_1_2.pdf
6.1.1 Release Notes	http://downloads.livecode.com/livecode/6_1_1/LiveCodeNotes-6_1_1.pdf
6.1.0 Release Notes	http://downloads.livecode.com/livecode/6_1_0/LiveCodeNotes-6_1_0.pdf
6.0.2 Release Notes	http://downloads.livecode.com/livecode/6_0_2/LiveCodeNotes-6_0_2.pdf
6.0.1 Release Notes	http://downloads.livecode.com/livecode/6_0_1/LiveCodeNotes-6_0_1.pdf
6.0.0 Release Notes	http://downloads.livecode.com/livecode/6_0_0/LiveCodeNotes-6_0_0.pdf