LiveCode 8.1.0-dp-2 Release Notes

Overview

LiveCode 8.1 provides important improvements for delivering high-quality cross-platform applications:

- The standalone builder now has a greatly-improved user experience for including externals, script libraries and LiveCode Builder extensions in your cross-platform application.
- The player control can be used in Windows application without any need for users to install any additional libraries or dependencies.

LiveCode 8.1 also contains several important upgrades for LiveCode Builder as part of the crowdfunded "LiveCode Infinity" project.

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 or the revVideoGrabber external.

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 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)
- Windows 10

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

Linux

LiveCode supports Linux installations which meet the following requirements:

- Supported CPU architectures:
    - 32-bit or 64-bit Intel/AMD or compatible processor
    - 32-bit ARMv6 with hardware floating-point (e.g. RaspberryPi)
- Common requirements for GUI functionality:
    - GTK/GDK/Glib 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)
- Requirements for 32-bit Intel/AMD:
    - glibc 2.11 or later
- Requirements for 64-bit Intel/AMD:
    - glibc 2.13 or later
- Requirements for ARMv6:
    - glibc 2.7 or later

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:

- 10.6.x (Snow Leopard) on Intel

- 10.7.x (Lion) on Intel
- 10.8.x (Mountain Lion) on Intel
- 10.9.x (Mavericks) on Intel
- 10.10.x (Yosemite) on Intel
- 10.11.x (El Capitan) on Intel

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 4.6 on MacOS X 10.7
- Xcode 5.1 on MacOS X 10.8
- Xcode 6.2 on MacOS X 10.9
- Xcode 6.2 and 7.2 on Mac OS X 10.10
- Xcode 7.3 on MacOS X 10.11

It is also possible to set other versions of Xcode, to allow testing on a wider range of iOS simulators. For instance, on Yosemite, you can add *Xcode 5.1* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 7.1*.

We currently support the following iOS Simulators:

- 6.1
- 7.1
- 8.2
- 9.2
- 9.3

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.

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 yet 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 noui [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
	Install the IDE for "All Users". If not specified, LiveCode will be installed

Option	Description
`-allusers`	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 noui [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 noui
```

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
```

Engine changes

Changes to the dontUseQT property of a player object (Windows and OSX) (8.1.0-dp-2)

It is now possible to set the **dontUseQT** property for a player object.

On Windows, the default value of the global **dontUseQt** and **dontUseQtEffects** properties has changed from true to false. This means that by default players created on Windows will use the DirectShow API for multimedia playback.

On OSX, QuickTime is unable to be supported in 64 bit builds the default value of the global **dontUseQT** and **dontUseQTEffects** properties changed in version 6.7; it is true on OS X version 10.8 and up, or on all versions of OS X if the engine is 64 bit. This means that any player object created will use the AVFoundation API for multimedia playback.

With this new feature, you can set the **dontUseQT** property of a player to false, without changing the value of the global **dontUseQt** property. If you do this, you can have both QuickTime and AVFoundation players playing at the same time, which can be particular useful for supporting some media formats or codecs that are not supported by the default AVFoundation or DirectShow player (for example .midi files, Sorenson Video 3, H.261 codecs etc)

Warning: QuickTime has not been maintained or supported by Apple for quite some time. You are encouraged to check your applications for any dependence on QuickTime, and remove it if found.

Add several IDE messages (8.1.0-dp-2)

Messages are now sent when audio clip and video clip controls are created or deleted.

- **newAudioclip**: sent when an audio clip is created
- **deleteAudioclip**: sent when an audio clip is deleted
- **newVideoclip**: sent when a video clip is created
- **deleteVideoclip**: sent when a video clip is deleted

Improved return command (8.1.0-dp-2)

The 'return' command has had two new forms added:

```
return value <value>
return error <value>
```

When running in a command handler, the 'return value' form will cause execution of the handler to halt, and control to return to the calling handler. At this point the 'it' variable in the calling handler will be set to 'value' and 'the result' will be set to empty. In contrast, the 'return error' form will cause the 'it' variable in the calling handler to be set to empty and 'the result' to be set to 'value'.

When running in a function handler, the 'return value' form will cause execution of the handler to halt, and control to return to the calling handler. At this point the return value of the function call will be 'value', and 'the result' will be set to empty. In contrast, the 'return error' form will cause the return value of the function call to be empty, and 'the result' will be set to 'value'.

These forms of return are designed to be used by script library functions to allow them to have the same ability as built-in engine functions and commands - namely the ability to return a value (in it for commands, or return value for functions) *or* return a status (in the result).

Revert a target stack (8.1.0-dp-2)

It is now possible to revert a stack that is not the topStack, using

```
revert <stack reference>
```

New vectorDotProduct function (8.1.0-dp-2)

A new **vectorDotProduct** function has been added. It computes the vector dot product of two single-dimensional arrays with identical keys.

More specifically:

```
vectorDotProduct(tArray1, tArray2)
```

Will compute:

```
put 0.0 into tSum
repeat for each key tKey in tArray1
    add tArray1[tKey] * tArray2[tKey] to tSum
end repeat
return tSum
```

If the two arrays do not have the same set of keys, then an error is thrown.

Add functions for getting synchronous modifier key state (8.1.0-dp-2)

LiveCode currently provides functions for checking the state of so-called "modifier" keys: Caps Lock, Control, Command, Shift, Alt/Option. These functions return either "up" or "down", reflecting the state of the key at the time the function was called. However, it is often desireable to check the state of the key at the time the event was generated and this is not possible using these functions.

New functions called "eventAltKey", "eventShiftKey", etc have been added; these return the state of the key at the time the event began processing. This is useful in keyDown and rawKeyDown handlers to check whether a modifier was pressed at the time the key the event relates to was pressed (if the non-event forms are used instead, there is a chance the modifier key has been released and the wrong result will be generated).

Note that the "eventXXXKey" functions should *not* be called after a wait; their value is undefined after any form of wait has occurred.

Curly brace subscripts are now a syntax error (8.1.0-dp-2)

Previously, it was possible to use curly brackets or braces `{}` instead of square brackets `[]` in array and custom property syntax. This was an undocumented and unknown feature to most users.

Using `{}` to subscript arrays is now a script syntax error. Curly brackets have been reserved for future use.

Add support for custom entitlements for iOS (8.1.0-dp-2)

Custom entitlements can now be added to an iOS app by including one or more `.xcent` files in the copy files section of the standalone builder containing an XML snippet of key/value pairs. For example, if you wanted to add the entitlement for HomeKit to your app you might create a file named `HomeKit.xcent` with the following content:

```
<key>com.apple.developer.homekit</key>
<true/>
```

Improve Android timestamp accuracy for GPS and sensors (8.1.0-dp-2)

Timestamps for sensors on Android were previously passed in a low-precision format, resulting in "sticky" timestamps that did not change more than a few times a minute. This has now been resolved and timestamps are now reported to microsecond resolution (though the accuracy is unlikely to be at the microsecond level).

In addition to this change, the timestamps are now reported in "monotonic" time rather than "wall-clock" time ("wall-clock" time is the time you see reported as the current time). This means that the timestamps are now independent of changes to the device clock as a result of adjustments or daylight savings changes. If you want to match the readings to the device time instead, get the current time when receiving the location update rather than using the timestamp in the update.

Improved GPS support on Android and iOS (8.1.0-dp-2)

GPS behavior is now identical on Android and iOS. On both platforms, the location reading returned by the `mobileSensorReading` function is that which was sent with the last system `locationChanged` event. (This brings iOS behavior inline with that of Android).

Additionally three new handlers have been implemented:

```
mobileGetLocationHistory
mobileSetLocationHistoryLimit
mobileGetLocationHistoryLimit
```

Whenever a system `locationChanged` event occurs, the location reading is pushed onto the front of a list. The list is capped at the length set by the location history limit, dropping any old samples over this length.

The `mobileGetLocationHistory` function returns a numerically keyed array of all accumulated samples since the last time it was called with lower indices being older samples. Calling the function clears the internal history.

Each element in the array is the same format as the detailed location array as returned from the `mobileSensorReading` function.

If an application wants historical access to all samples, then it should set the location history limit to the maximum number of samples it ever wants to record, or 0 to record the entire history (between calls to `mobileGetLocationHistory`).

The best way to use the history is to fetch the list in `locationChanged` and process each sample in turn, rather than the sample provided with the `locationChanged` event (which will always be the last sample in the history). e.g.

```
on locationChanged
   local tHistory
   put mobileGetLocationHistory() into tHistory
   repeat for each element tSample in tHistory
      processLocationChanged tSample
   end repeat
end locationChanged
```

The default history limit is 1 meaning that only one sample is ever kept at a time.

Automatic LCB extension inclusion in standalones (8.1.0-dp-1)

When a standalone is built, the modules required for the widgets that are on the stack (or any of its substacks) are now included in the application automatically, regardless of whether the 'Search for required inclusions...' option is selected in the standalone settings.

If 'Search for required inclusions' is enabled, the scripts of the application will be searched for uses of the public handlers of any available LCB libraries, and any uses of the 'kind' of available widgets to determine whether the relevant modules are included. For example, if the script contains:

```
create widget as "com.livecode.widget.svgpath"
```

then the 'SVG Path' widget and all its dependencies will be included.

Standalone 'Search for inclusions' for mobile deployment (8.1.0-dp-1)

The standalone builder 'Search for required inclusions...' option now supports mobile deployment, both to device and simulator.

Standalone script library inclusions for mobile deployment (8.1.0-dp-1)

Script libraries can now be included in mobile applications in the same way as for desktop applications, via the 'Inclusions' pane of the standalone builder.

Windows DirectShow Player Control (8.1.0-dp-1)

Due to the recent decision by Apple to end support for QuickTime on Windows, the player implementation on that platform has been replaced with one based on DirectShow. This is a multimedia API that is available by default on all versions of Windows supported by LiveCode.

The new implementation should function as a drop-in replacement for the old one, though some properties are not yet implemented.

Property Changes

On Windows, the behaviour of some properties of the player control have changed.

- The **loadedTime** property previously did not work on Windows, but now does.
- The **alwaysBuffer**, **enabledTracks**, **mediaTypes**, **mirrored**, **trackCount** and **tracks** properties do not currently work, but will be re-enabled in a subsequent release.

On all platforms, the following player control properties, which are specific to QuickTime and QTVR, have been deprecated: **constraints**, **currentNode**, **movieControllerId**, **nodes**, **pan**, **tilt**, and **zoom**.

Supported File Formats

Media format support in the new Windows player control depends on which codecs are installed.

A list of the file formats and compression types available as standard.aspx) on Windows is available in the MSDN documentation

Specific engine bug fixes (8.1.0-dp-2)

15183	**Ensure dependencies of built-in inclusions are included**
15366	**Correctly update stack rectangle when moving to a different screen**
17180	**Ensure deleted objects executing scripts can not be deleted**
17275	**Add functions for getting synchronous modifier key state**
17317	**Deprecate liveResizing and metal stack properties**
17469	**Curly brace subscripts are now a syntax error**
17515	**Add support for custom entitlements for iOS**
17523	**Fix LCB docs builder handling of string-like property names**
17553	**Paint Tools Not Working in IDE**
17571	**Fix PDF display in CEF-based browser widget (Windows, Linux)**
17573	**Don't retain other platforms' temporary standalone settings data**
17609	**Return an empty item instead of a random value if altitude reading is not available on iOS**
17620	**Fix javascript handlers of browser widget not callable on Android**
17637	**Update docs to reflect changes to standalone builder inclusions**
17661	**Improve Android timestamp accuracy for GPS and sensors**
17662	**Improved GPS support on Android and iOS**
17697	**Fix player view occasionally not showing on Mac**
17698	**Fix Windows player frame seeking**
17700	**Fix Windows player not pausing when in edit mode**
17701	**Fix incorrect Windows player rect on opening**
17708	**Fix incorrect player currenttime value for videos longer than 7m 15s**
17720	**Document MetaCard compatible pattern numbers**
17725	**Add support for pattern numbers to backdrop**
17747	**Make sure widgets get mouseUp in popup stacks**
17754	**Ensure external code blobs are included in standalones**
17776	**Parse deprecated LCB syntax properly**

17781	**Fix OSX mouse event errors when using QTKit player**
17797	**Enable playback of MP3 and other audio files in Windows player**
17800	**Ensure all parameters are included when using send script**
17802	**Document how to escape special characters in wildcard filter patterns**
17815	**Fix native layer of player not showing when stack opened**
17828	**Fix player slider moving outside its track boundary**
17834	**Prevent possible assertion failure related to revMessageBoxRedirect**
17839	**Corrected mistaken key name in Info.plist file on iOS standalone**
17842	**Ensure read from socket documentation matches engine behavior**
17844	**Allow the dropChunk function to parse as a chunk reference**
17856	**Deal with pdfPrinter inclusion setting correctly**
17891	**Reset the m_was_licensed instance variable to true before calls to an external's handler**
8212	**Correct mapping of pattern number to image id**
9778	**Fix 'cut tVar' where tVar contains an object text chunk**

IDE changes

Deprecated syntax in the dictionary (8.1.0-dp-2)

Deprecated syntax is now be prefixed in the dictionary entry list by a warning icon.

Dictionary UI Improvements (8.1.0-dp-2)

The dictionary stack UI has been updated to incorporate a few user interface improvements:

- The panel containing list of entries that match the current filter and search terms is now resizable, and contains three columns - the name of the entry, the type, and the syntax. The list can be sorted by any of these three.
- The history breadcrumb has been condensed into back and forward buttons, plus a history dropdown menu button.
- The filters pane now has associations and platforms filter categories, and is now scrollable if the content is too large for the pane. The numbers associated with each filter have been removed. The list of filters is laid out in two columns, sorted in alphabetical order.

Autofocus on message box when typing (8.1.0-dp-2)

If the message box is open, it will now automatically gain focus and receive keystrokes when you start typing with no field focused. This makes it much faster and more convenient to quickly run message box commands without hunting for the message box.

Standalone Inclusions Interface (8.1.0-dp-1)

The standalone settings user interface has been reworked to unify the notion of app inclusion. There is now an 'Inclusions' pane which allows the user to select from a complete list of available inclusions. The list contains information about which platforms are supported.

The 'Inclusions' pane significantly improves the cross-platform development experience provided by LiveCode (since the iOS and Android panes no longer have separate check boxes for the various built-in externals such as revxml), as well as paving the way for much better extensibility in the future.

Specific IDE bug fixes (8.1.0-dp-2)

16167	**Remove reference to Windows XP from preferences stack**
16325	**Ensure preferences reset correctly**
17146	**Ensure that the Script Editor "Find" UI resizes correctly**
17189	**Replace Revolution text with LiveCode in Dialogs**
17405	**Display virtual font names correctly in the script editor font preference menu**
17467	**Fix recent files list update issue**
17739	**Make object list in project browser flat when sorting by name**
17759	**show alpha values of gradients dynamically**
17826	**Make sure "move" command results in smooth movement when executed from the msg box**
296	**Autofocus on message box when typing**
8985	**Clicking type disambiguation in Docs pane of Script Editor should work**

LiveCode Builder changes

LiveCode Builder Language

Variables

- Out parameters are now initialized by default to a suitable empty value at the start of the handler. For example:

  ```
  public handler GetMyValue(out rValue as Integer)
  end handler
  ```

  will result in `rValue` being set to 0.

- Handler local variables are now have lexical scope. This means variables are accessible from the point of definition to the end of the block they are defined in. Note that:

  - each `repeat` control structure is considered a single block.

  - each separate block in an `if`/`else if`/`else` control structure is considered a single block.

- Variables are now reset (either to their default value, or unassigned) at the point of the variable definition. In particular, any variables defined within a `repeat` block are reset on each iteration:

```
repeat 5 times
   variable tVar as optional String
   -- tVar is reset to "nothing" every time the loop runs
end repeat
```

- Variables in an inner block can now shadow those in an outer block. For example, the following is valid:

```
variable tX as Array
repeat 5 times
   variable tX as Number
   repeat 4 times
      variable tX as String
   end repeat
end repeat
```

- Typed variables are now initialised by default to a suitable empty value. For example:

```
variable tList as List
push "element" onto tList
```

  Untyped and `optional` variables are initialised to `nothing`.

Bytecode Blocks

- bytecode can now be directly written in handlers using a bytecode block:

```
bytecode
   register tTemp
   assign_constant tTemp, 1
end bytecode
```

- for more details on what bytecode operations can be used see the LiveCode Builder Bytecode Reference
- bytecode blocks are not for general use and the current set of bytecode operations are subject to change

Unsafe Attributes

- The compiler now understands the idea of 'safety' of handlers and blocks of code.

- Handlers can be marked as being 'unsafe', e.g.

```
unsafe handler Foo()
    ... do unsafe things ...
end handler
```

- Blocks of statements can be marked as being 'unsafe', e.g.

```
unsafe
    ... do unsafe things ...
end unsafe
```

- All foreign handlers are considered to be 'unsafe'.
- All bytecode blocks are considered to be 'unsafe'.
- Calls to foreign handlers and unsafe handlers can only be made within unsafe handlers or unsafe statement blocks.
- Usage of bytecode blocks can only be made within unsafe handlers or unsafe statement blocks.

Specific LCB bug fixes (8.1.0-dp-2)

16212	**Escape XML reserved characters in manifest files**
17526	**Make variables lexically scoped in statement blocks.**
17767	**Process escape sequences in all string literals.**
17809	**Initialize out parameters to default.**

Specific LCB bug fixes (8.1.0-dp-1)

14659	Initialise variables with a default value

LiveCode extension changes

Line Graph widget

Properties

- Throw an error when the **graphData** property is set to an invalid value, such as an empty string.

SVG Icon widget

Properties

- New **scaledWidth** and **scaledHeight** properties have been added. These are read-only properties that expose the effective size of the rendered SVG path, independent of the widget's size.

  When **maintainAspectRatio** is false, these values are equal to the width and height of the widget.

Specific extension bug fixes (8.1.0-dp-2)

17373	**enable setting local timezone, make local default**

Specific extension bug fixes (8.0.2-rc-1)

17692	Prevent errors in onPaint with empty graphData

Dictionary additions

- **deleteAudioclip** (*message*) has been added to the dictionary.
- **deleteVideoclip** (*message*) has been added to the dictionary.
- **eventAltKey** (*function*) has been added to the dictionary.
- **eventCapsLockKey** (*function*) has been added to the dictionary.
- **eventCommandKey** (*function*) has been added to the dictionary.
- **eventControlKey** (*function*) has been added to the dictionary.
- **eventOptionKey** (*function*) has been added to the dictionary.
- **eventShiftKey** (*function*) has been added to the dictionary.
- **libURLSetDriver** (*command*) has been added to the dictionary.
- **mobileGetLocationHistory** (*function*) has been added to the dictionary.
- **mobileGetLocationHistoryLimit** (*function*) has been added to the dictionary.
- **mobileSetLocationHistoryLimit** (*command*) has been added to the dictionary.
- **newAudioclip** (*message*) has been added to the dictionary.
- **newVideoclip** (*message*) has been added to the dictionary.
- **vectorDotProduct** (*function*) has been added to the dictionary.

Previous 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 6.6.2 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