LiveCode 9.0.0-dp-9 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 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 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) on Intel
- 10.10.x (Yosemite) on Intel
- 10.11.x (El Capitan) on Intel
- 10.12.x (Sierra) 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 6.2 on MacOS X 10.9
- Xcode 6.2 and 7.2 on Mac OS X 10.10
- Xcode 8.2 on MacOS X 10.11
- Xcode 8.3 on MacOS 10.12

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
- 10.3

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.

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)
- 7.0 (Nougat)

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 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
`-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 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
```

LiveCode Community engine changes

LiveCode Builder changes

Java CLASSPATH support

Limited support is available for loading custom Java classes and .jar files in the IDE on Mac and Linux. If the CLASSPATH environment variable is set before the Java virtual machine is initialised (i.e. before Java FFI is used), then any paths specified are added to the locations searched by the default class loader.

HTML5 Networking Support (Updated 9.0.0-dp-9)

Networking support has been updated in the HTML5 engine to support libURL like syntax for fetching HTTP and HTTPs URLs. In order to use this new functionality, make sure "Internet" is selected in the list of inclusions.

Note: URLs fetched by the HTML5 engine from a domain other than that of the hosting the page may be blocked by web browsers, unless the server hosting the URL sets the "Access-Control-Origin" header appropriately.

Add support for accepting socket connections on a port in the ephemeral port range

When accepting connections on port 0 the OS will assign an available port within it's ephemeral port range. The accept command now names the socket with the bound port number rather than 0 so that the bound port will appear in the value of the openSockets function and sets the it variable to the port number.

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) on Intel
- 10.10.x (Yosemite) on Intel
- 10.11.x (El Capitan) on Intel
- 10.12.x (Sierra) 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 6.2 on MacOS X 10.9
- Xcode 6.2 and 7.2 on Mac OS X 10.10
- Xcode 8.2 on MacOS X 10.11
- Xcode 8.3 on MacOS 10.12

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
- 10.3

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.

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)
- 7.0 (Nougat)

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.

New array commands `difference` and `symmetric difference`

The `difference` command removes all keys from the destination which are present in the source, and leaves all others alone.

The `symmetric difference` command removes all keys from the destination which are present in the source, and adds all keys from the source which are not present in the destination.

Additionally the `into` clause has been added to all array set set operations (`union`, `intersect`, `difference`, `symmetric difference`) allowing commands such as:

intersect tLeft with tRight into tResult

The operation of the commands is the same as the non-into form except that tLeft does not have to be a variable, and the result of the operation is placed into tResult rather than mutating tLeft.

New messageDigest() function with SHA-2 and SHA-3 support

A new `messageDigest()` function has been added. It allows access to a variety of cryptographic message digest functions, including SHA-2 and SHA-3. For example, to compute the 256-bit SHA-3 digest of the message "LiveCode", you might use:

```
get messageDigest(textEncode("LiveCode", "UTF-8"), "sha3-256")
```

Update Skia (graphics library)

This major Skia update improves both rendering quality and performance. It also opens the door to substantial future improvements and feature additions. In order to allow this, support for certain legacy drawing features has had to be removed. In particular, legacy blend modes (also known as bitmap effects) are no longer supported.

Cmd-. does not affect modal dialogs

Previously using the abort-script keyboard combination (Cmd-. on Mac) would cause an abort error to be thrown. However, this would be silently swallowed by any modal command (or equivalent) meaning that unusable modal dialogs would be uncloseable, requiring the need to restart the IDE / engine.

This has been fixed by making Cmd-. cause an automatic 'close this stack' when it occurs in a modal loop and allowInterrupts is true, and the current stack has cantAbort set to false.

Additional forms of create command

- Create in now works correctly
- You can now create in as well as in

recordFormats function

A new `recordFormats` function has been added that lists the possible record formats supported by the current platform recorder implementation. This will allow users to select supported audio file formats used by 'pluggable' audio recorder implementations in the future.

LiveCode Builder Tools

lc-compile-ffi-java

- **lc-compile-ffi-java** is a tool to create LiveCode Builder code which provides an interface between LCB modules and the Java Native Interface (JNI). It does this by taking a specification of a java package written in a domain-specific language and creating an LCB module which wraps the necessary foreign handlers which call the appropriate JNI methods. See the documentation for the tool and the description of the DSL for more details

lc-compile

Errors

- A new error has been added for identifiers in declaration context that contain `.` - identifiers should always be declared without qualification.
- Parsing of numeric literals, in general, has been tightened up. In particular, the compiler will detect invalid suffixes on numeric literals meaning you cannot accidentally elide a number with an identifier.

```
1.344foo -- ERROR
0xabcdefgh -- ERROR
0b010432 -- ERROR
```

Messages

- Errors, warnings and informational messages now display the affected line of code and visually indicate the position where the problem was found. For example, the output might look like:

```
foo.lcb:2:26: error: Identifier 'InvalidExpression' not declared
  constant kBadConstant is InvalidExpression
                           ^
```

lc-run

- **lc-run** now has the *experimental* ability to load and run bytecode assemblies containing multiple LCB modules. To construct a multi-module bytecode assembly, simply concatenate several `.lcm` module files together. The first module found in a bytecode assembly is treated as its main module.

LiveCode Builder Language

Foreign bindings to Java

It is now possible to bind to Java class methods and fields in LCB via the Java Native Interface (JNI). Foreign handlers bind to Java class using the existing foreign handler syntax, but with an appropriate binding string.

The Java binding string has the following form:

```
"java:[className>][functionType.]function[!calling]"
```

Where *className* is the qualified name of the Java class to bind to, *functionType* is either empty, or 'get' or 'set', which are currently used for getting and setting member fields of a Java class, and *function* specifies the name of the method or field to bind to. The function 'new' may be used to call a class constructor. *function* also includes the specification of function signature, according to the standard rules for forming these when calling the JNI.

calling specifies the calling convention which can be one of:

- `instance`
- `static`
- `nonvirtual`

Instance and nonvirtual calling conventions require instances of the given Java class, so the foreign handler declaration will always require a Java object parameter.

Examples

- Binding to a class constructor with no parameters:

  foreign handler CreateJavaObject() returns JObject binds to "java:java.lang.Object>new()"

- Binding to a class constructor with parameters:

  foreign handler CreateJavaString(in pBytes as JByteArray) returns JString binds to "java:java.lang.String>new([B)"

- Binding to a class instance method

  foreign handler JavaStringIsEmpty(in pString as JString) returns CBool binds to "java:java.lang.String>isEmpty()Z"

- Binding to a class static method

  foreign handler CallJavaAdd(in pLeft as CInt, in pRight as CInt) returns CInt binds to "java:java.lang.Math>addExact(JJ)J!static"

- Binding to a class field

  foreign handler JavaCalendarSetTime(in pObj as JObject) returns nothing binds to "java:java.util.Calendar>set.time(J)" foreign handler JavaCalendarGetTime(in pObj as JObject) returns CInt binds to "java:java.util.Calendar>get.time()J"

- Binding to a class constant

  foreign handler GetJavaPi() returns CDouble binds to "java:java.lang.Math>get.PI()D!static"

  Note: This feature is still highly experimental. All aspects of the syntax are subject to change. Incorrect binding strings may cause your application to crash, so please ensure you back up your stacks before experimenting.

  Important: This feature is currently supported on Android, Mac and Linux. Binding to java classes requires the availability of a Java runtime and access to the appropriate libraries. On Mac, the `JAVA_HOME` environment variable must be set to the path to your Java installation (usually at

`/Library/Java/JavaVirtualMachines/jdk1.7.0_55.jdk/Contents/Home`). On Linux, your `LD_LIBRARY_PATH` must be set to the folder containing the `libjvm.so` library (usually at ${JAVA_HOME}/jre/lib/amd64/server) on 64-bit Linux.

Foreign Function Interface

- Obj-C exceptions thrown from calls to obj-c foreign handlers will now be caught and rethrown as an LCB error.
- It is now possible to specify the thread to be used in Obj-C foreign handlers. This is done by appending `?<thread>` to the end of the binding string. Currently the only supported value is `ui`, for running the handler on the iOS main thread, as opposed to the engine thread.
- It is now possible to specify the thread to be used in Java-bound foreign handlers. This is done by appending `?<thread>` to the end of the binding string. Currently the only supported value is `ui`, for running java on the android UI thread. This has no effect on desktop platforms.

Android Listener support

The interface callback return value should now match that of the invoked handler, i.e. if it returns void, the LCB handler should return nothing, otherwise it should return JObject.

An binding string variant has been added which allows the user to create an interface proxy - that is an instance of a generic Proxy class for a given interface.

This effectively allows LCB handlers to be registered as the targets for java interface callbacks, such as event listeners.

The syntax is as follows:

```
foreign handler _JNI_CreateListener(in pMapping) returns JObject binds to
"java:listener.class.path>interface()"
```

The foreign handler binding to such a function takes a value that should either be a `Handler` or an `Array` - if it is a `Handler`, the specified listener should only have one available callback. If the listener has multiple callbacks, an array can be used to assign handlers to each. Each key in the array must match the name of a callback in the listener. Overloaded methods in the interface are not currently supported.

For example:

```
handler type ClickCallback(in pView as JObject)

foreign handler _JNI_OnClickListener(in pHandler as ClickCallback) returns
JObject binds to "java:android.view.View$OnClickListener>interface()"

foreign handler _JNI_SetOnClickListener(in pButton as JObject, in pListener as
JObject) returns nothing binds to
"java:android.view.View>setOnClickListener(Landroid/view/View$OnClickListener;)V"


public handler ButtonClicked(in pView as JObject)
    -- do something on button click
end handler

public handler SetOnClickListenerCallback(in pButton as JObject)
    unsafe
        variable tListener as JObject
        put _JNI_OnClickListener(ButtonClicked) into tListener
        _JNI_SetOnClickListener(pButton, tListener)
    end unsafe
end handler
```

or

```
handler type MouseEventCallback(in pMouseEvent as JObject)

foreign handler _JNI_MouseListener(in pCallbacks as Array) returns JObject binds
to "java:java.awt.event.MouseListener>interface()"

foreign handler _JNI_SetMouseListener(in pJButton as JObject, in pListener as
JObject) returns nothing binds to
"java:java.awt.Component>addMouseListener(Ljava/awt/event/MouseListener;)V"

public handler MouseEntered(in pEvent as JObject)
    -- do something on mouse entter
end handler

public handler MouseExited(in pEvent as JObject)
    -- do something on mouse entter
end handler

public handler SetMouseListenerCallbacks(in pJButton as JObject)
    variable tArray as Array
    put MouseEntered into tArray["mouseEntered"]
    put MouseExited into tArray["mouseExited"]
    unsafe
        variable tListener as JObject
        put _JNI_MouseListener(tArray) into tListener
        _JNI_SetMouseListener(pJButton, tListener)
    end unsafe
end handler
```

Foreign aggregate support (experimental)

- It is now possible to bind to C functions and Obj-C methods which take simple structs by-value, or return a struct by-value. For full details see the 'Foreign Aggregate Types' section of the language reference.

Variadic foreign C functions

- It is now possible to bind to variadic C functions: foreign handler printf(in pFormat as Pointer, ...) returns CInt binds to "" In this case, the '...' must be the last parameter, and there must be at least one fixed parameter.

Identifiers

- Unqualified identifiers are now expected to match `[A-Z0-9_]`. The `.` symbol is interpreted as a namespace operator.

Namespaces

- Identifiers in LCB can now be qualified, to disambiguate between symbols defined in different namespaces.

Literals

- Base 2 (binary) integer literals can now be specified by using a "0b" prefix, e.g.

```
0b0000
0b1010
```

- Base 16 (hexadecimal) integer literals can now be specified by using a "0x" prefix. e.g.

```
0xdeadbeef
0x0123fedc
```

Message box refactor

The way the message box functions has been refactored:

- the IDE only global property `revMessageBoxRedirect` has been removed
- the IDE only global property `revMessageBoxLastObject` has been removed
- the legacy message box behavior setting the text of the first field of a stack named `Message Box` has been removed
- the `msgChanged` message is now sent to the object that changed the message
- IDE plugin developers should subscribe to `ideMsgChanged` for custom message box development.
- If the `msgChanged` message is not handled the content of the `message box` will be logged to the system log unless the engine is running in no ui (command line) mode which will write the content to STDOUT.

revvideograbber end-of-life on Mac OS X

The revvideograbber external depends on QuickTime in order to operate on Mac OS X. On Mac OS X 10.9 and above, QuickTime is no longer available.

The revvideograbber external remains available for use on Windows, using Video for Windows (DirectShow).

The following revvideograbber commands are now deprecated and have no effect:

- **revVideoGrabSettings**
- **revSetVideoGrabSettings**
- **revVideoGrabIdle**

New NFC tag feature on Android

It is now possible to access data directly from NFC tags read by Android devices.

The functions **mobileIsNFCAvailable** and **mobileIsNFCEnabled** can be used to test if the device is able to read NFC tags.

The commmands **mobileEnableNFCDispatch** and **mobileDisableNFCDispatch** can be used to control whether or not your app intercepts all NFC tags while in the foreground.

The handler **nfcTagReceived** will be sent with the tag data to your app whenever an NFC tag is read by the device.

LCB modules can declare Android app permissions and features

LCB module metadata is now checked for Android permissions which will be added to the manifest when building for Android. For example, a module containing

```
metadata android.features is "hardware.bluetooth,hardware.camera"
metadata android.hardware.camera.required is "false"
metadata android.hardware.bluetooth.required is "true"
metadata android.permissions is "BLUETOOTH_ADMIN"
```

will result in the following lines being added to the Android manifest:

```
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.bluetooth" android:required="true"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

RevFont has been removed

The revFont external library has been removed as it is no longer required.

The two commands provided by the revFont library have been superceded by commands that are built into the LiveCode engine. The replacements are:

- `revFontLoad` -> `start using font`
- `revFontUnload` -> `stop using font`

Specifying local host and port when opening a socket

When opening a socket you can now specify the local port and host the socket should use. An example of this is as follows:

```
open socket from ":8080" to "10.2.1.1:8080"
```

See the open socket dictionary entry for full details.

Maximum text length on iOS native input fields

It is now possible to set/get the maximum number of characters that can be entered into an ios native single-line field, using

```
mobileControlSet sFieldID, "maximumTextLength", sMaxLength
put mobileControlGet(sFieldID, "maximumTextLength") --> returns sMaxLength
```

Support for loading multi-module bytecode files (experimental)

The **load extension** command is now able to load LiveCode Builder bytecode files (`.lcm` files) that contain multiple modules' bytecode.

The first module in each `.lcm` file is treated as the "main module" of the module (i.e. the library or widget), and other modules are treated as support modules.

Support modules only remain loaded if they are used by the main module, and support modules must be submodules of the main module. For example, if the main module is "com.livecode.newbutton", then all other modules in the bytecode file must have names like "com.livecode.newbutton.<something>".

Throw error when changing behavior from behavior script

Previously it was theoretically possible to change the behavior of an object from that object's existing behavior script. This will now result in an execution error

```
parentScript: can't change parent while parent script is executing
```

This change was necessarily as the engine would occasionally crash when changing a behavior this way, and would be guaranteed to crash if stepping over the behavior script line that changes the behavior.

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

Re-written LCB VM

The "virtual machine" used to run LiveCode Builder code has been re-written from scratch. This new VM provides a framework enabling better extensibility, better error reporting and, in future, more comprehensive optimizations.

Most existing LCB code should run without any changes. There may be some code that worked on the previous VM but doesn't in the new VM due to more comprehensive run-time checking; this is usually fixable with only very minor changes to the source code.

Calling JavaScript from HTML5

JavaScript has been added to the **alternateLanguages** on the HTML5 platform.

It is now possible to call JavaScript code from HTML5 standalones by using the `do <script> as <alternateLanguage>` form of the **do** command.

This allows HTML5 standalones to interact with the browser within which they are running. The value of the JavaScript expression will be placed in the **result** variable:

```
local tDocTitle
do "document.title" as "JavaScript"
put the result into tDocTitle
```

Undocumented multi-file libUrlMultipartFormAddPart removed

Previously, the **libUrlMultipartFormAddPart** command had the undocumented capability to accept multiple file names separated by commas. The handler failed to work for files that had commas in the name, however. The undocumented behaviour has been removed. To add multiple files to a form, call **libURLMultipartFormAddPart** once for each file.

Field tab alignments in htmlText and styledText

The **styledText** and **htmlText** of a field now include tab alignment information. The **htmlText** uses a new `tabalign` attribute with a list of alignments, e.g.

```
<p tabalign='left,center,right'>left&09;middle&09;right&09;</p>
```

The **styledText** stores tab alignment in a "tabalign" key in each paragraph's "style" array, e.g.

```
get tStyledText[1]["style"]["tabalign"]
```

Platform support end-of-life

As announced on the LiveCode blog, running LiveCode on the following platforms is no longer officially supported from LiveCode 9.0 onwards:

- Windows XP
- Windows Server 2003
- Windows Vista
- Android Gingerbread (2.3.3-2.3.7)
- Android Ice Cream Sandwich (4.0)
- OS X Snow Leopard (10.6)
- OS X Lion (10.7)
- OS X Mountain Lion (10.8)
- iOS Simulator 6.1
- iOS Simulator 7.1

LiveCode Builder Standard Library

Foreign function interface

- The Java integer primtive type JChar has been added. This maps to uint16_t.
- The machine types Bool, SIntSize, UIntSize, SIntPtr, UIntPtr, SInt8, UInt8, SInt16, UInt16, SInt32, UInt32, SInt64 and UInt64 have been added. These all map to their corresponding foreign counterparts.
- The C types CBool, CChar, CSChar, CUChar, CSShort, CUShort, CSInt, CUInt, CSLong, CULong,

CSLongLong and CULongLong have been added. These all map to their corresponding C counterparts.

- The Java integer primtive types JBoolean, JByte, JShort, JInt, JLong, JFloat and JDouble have been added. These map to int8_t, int16_t, int32_t, int64_t, float and double.

Sequence operations

- New syntax has been added for reversing the contents of sequence types (`List`, `String` and `Data`). The `reverse <Value>` statement reverses the order of the sequence.

Java Utilities

Syntax for converting between a JObject and Pointer have been added to the Java utility library.

- `PointerToJObject` - converts a Pointer to a JObject
- `PointerFromJObject` - converts a JObject to a Pointer

These can be used in APIs where `Pointer` is the type of a platform-agnostic parameter whose underlying type is assumed to be `jobject` when used in a platform-specific implementation, for example:

```
-- pView is assumed to be a JObject with underlying type android.view.View
set my native layer to PointerFromJObject(pView)
```

There is now a utility library for manipulating java objects. It contains a type `JObject` which wraps a Java object, some type conversion operations:

- `StringFromJString` - converts a Java string to an LCB String
- `StringToJString` - converts an LCB String to a Java string
- `DataFromJByteArray` - converts a Java byte array to LCB Data
- `DataToJByteArray` - converts LCB Data to a Java byte array

and a utility for determining the class of a given java object:

- `GetJavaClassName` - return an LCB String containing the class name of the given `JObject`

Assertions

- Checks for handler preconditions can now be included using the new `expect` statement. For example, the following statement will throw an error if the value `pProbability` is out of the valid range for probabilities:

  ```
  expect that (pProbability >= 0 and pProbability <= 1) \
     because "Probabilities must be in the range [0,1]"
  ```

  The `that` keyword and the `because <Reason>` clauses are optional.

LiveCode Builder Host Library

Engine library

- Widgets and library modules can now get the path to their resources folder using `my resources folder`. If there is no resources folder attached to the calling module, then nothing is returned.
- The `execute script` command has been improved and now allows specification of a target object and list of arguments. For example:

  variable tScriptObject as ScriptObject resolve script object "this card" into tScriptObject execute script "return param(1) + param(2)" in tScriptObject with [1, 2]
- Library handlers can now access the delimiter properties set in the most recent script context using `the line delimiter`, `the item delimiter`, `the row delimiter` and `the column delimiter`.

LiveCode Builder Documentation

Style guide

- Updated naming guide for handlers and types
- Added indentation and wrapping guidelines
- New section with widget-specific recommendations

Specific engine bug fixes (9.0.0-dp-9)

15924 **HTML5: unable to import URL**
17810 **Allow non-LF line endings in script only stacks**
19094 **Certain unicode characters render incorrectly on Android 7**
19496 **'compositorType' dict entry updated to refect platform OpenGL reference**
19894 **Fix incorrect rendering of carets in fields**
19901 **Statically link Linux server engines to libstdc++**
19926 **Fix incorrect rendering of legacy gradients when stack scaling is applied**
19973 **Allow bind to interface callbacks with non-void return**
19981 **Fix inability to use custom fonts on Windows**
20013 **Database type passed to revOpenDatabase should be case insensitive**
20016 **DataGrids completely unresponsive in HTML 5 standalones**
20035 **Fix incorrect fontnames listed on Windows**
20079 **Add `the square root of <number>` to the math library**
20085 **Fixed syntax errors examples in the httpdStart and httpdResponse dictionary entries**
20089 **Ensure obj-c action proxy objects are deallocated correctly**
20183 **Ensure `import paint into group` respects the parent group**
20218 **Fix incorrect rendering of mirrored & wrapped gradients**
20290 **Support PKCS#1 formatted public keys in the encrypt/decrypt commands**
20325 **Add support for foreign aggregates**
20325 **Add support for foreign aggregates**
20326 **"Ask file" dialog does not appear and eventually causes crash on Mac**
20347 **Fix crash when cloning groups**

20349 **Fix crash when deleting the focused object**
20357 **Ensure the points list is displayed correctly, without showning -32768,-32768 for empty line**
20363 **Update DIB format when image placed on Windows clipboard**
20410 **Script Editor can drop keystrokes on Windows**
20411 **Fix crash when using Obj-C FFI on iOS**
20413 **Add documentation for android specialFolderPath external prefix**

Specific engine bug fixes (9.0.0-dp-8)

11827 Added quotes to examples in the visual effect dictionary so that they work in Strict Compilation Mode.
18899 Fix load image from resource file
19783 Ensure any errors are cleared from the error stack after a try control structure
19871 Add tools palette icon for android native button
19877 Add support for accepting socket connections on a port in the ephemeral port range
19911 Ensure interface callbacks are executed on engine thread
19911 Ensure interface callbacks are executed on engine thread
19927 Allow ignorable whitespace after continuation chars
19931 Fix loading of shared libraries on Android
19945 Mac native layer snapshots are offset by 1 pixel
19974 Enable using test runner with awkward paths to engines
19991 Add support for variadic foreign C functions.
19991 Add support for variadic foreign C functions.
20078 Allow modules to access their resources folder

Specific engine bug fixes (9.0.0-dp-7)

11323 New array commands `difference` and `symmetric difference`
13102 Update currentTimeChanged message docs to state that the message is sent while player is playing.
14861 Add "reverse _" syntax for sequence types
17132 Update engine to use LCMS 2 on Linux
18754 Set the it when importing snapshots
18848 Ensure error whilst doing subwindow command propagates to caller
18922 Added mobileGetDeviceToken as related to some pushNotification* dictionary entries
19083 Added selectionChanged association to player object
19165 Tweak formating on choose command doc removing return in browse param to remove word fields from top of param list.
19253 Update OpenSSL to version 1.1.0e
19271 Increased random function input limit to 2^53
19280 Include core script-only libraries in standalones
19318 Fix loading custom properties from pre-7.0 stack formats
19349 Fix relayering by layer number
19399 Prevent crash on startup of emscripten engine
19400 Unmangle java module docs

19411 Make sure `combine tArray using column` works as expected
19422 Fixed missing parts of errorMode entry
19429 Fix issue with database driver inclusion in standalones on Desktop
19479 Cmd-. does not affect modal dialogs
19486 Failure to bind foreign handlers should provide meaningful error
19502 Make sure calling LCB library handlers produces correct error stack
19523 Make sure all created registers are destroyed
19598 Cannot accept and open sockets using the same local port on certain platforms
19682 Fixed bug preventing the use of shift to draw straight lines
19779 fix bytecode generation for 'set my native layer' syntax
19784 Convert between stringref and jstring correctly
19804 Ensure fonts work in-ui mode on Windows
19855 Added cross reference to trueWord to the word entry in the dictionary.

Specific engine bug fixes (9.0.0-dp-6)

17318 Fix format of mobileControlDo entry
19066 Improve LCB 'execute script' command.

Specific engine bug fixes (9.0.0-dp-5)

16044 Corrected documentation for dateItems - Day of the week
18245 Message box refactor
18403 Support hidden paragraph property in styledText arrays
18622 Ensure empty maps to empty lists when calling LCB library handlers
18850 LCB modules can declare Android app permissions and features
18998 Fix go url for script-only stacks
19046 Ensure error is reported for undeclared identifiers
19059 Add access to script delimiters
19065 Improve error reporting for calling LCB library handlers
19180 Crash when cloning an empty MCObjectPropertySet

Specific engine bug fixes (9.0.0-dp-4)

11039 Throw error when changing behavior from behavior script
18107 Do not permit namespace operator in unqualified identifiers.
18853 Support for loading multi-module bytecode files (experimental)
18929 Update LiveCode Builder ABI version for LiveCode 9

Specific engine bug fixes (9.0.0-dp-2)

12196 Correct documentation for "do" command
18147 The scriptExecutionErrors property not listed in dictionary
18350 Fix spurious type errors for repeat variables in LCB
18495 Undocumented multi-file libUrlMultipartFormAddPart removed

18651 Ensure "10 garbage" is never a number
18821 Report all LCB stack frames in LCS error info

Specific engine bug fixes (9.0.0-dp-1)

14645 Field tab alignments in htmlText and styledText
15865 Fixed Dictionary description for "is not among"
16211 Fix compilation errors with MacOSX SDK 10.10 and higher
18086 Improve and expand LCB style guide
18111 Make PDF user guide typography match dictionary view
18254 Improve efficiency of equality operators on binary data
18297 Broken references in "filename of stack" dictionary entry
18357 dispatch documentation should mention arguments can be arrays
18385 lc-run: Load multi-module bytecode assemblies.
18463 Show correct error position when source line includes tabs
18537 Fix crash when saving stack on OSX ElCapitan
18579 Support defaultNetworkInterface for the accept command
18588 Fix a crash due to pending messages to deleted objects

LiveCode Community IDE changes

The IDE is now 64-bit by default on Mac

Moreover, the "Build for Mac OS X 64-bit" is checked by default on newly created stacks in the standalone settings for OS X. Existing stacks will retain their current settings.

Added "Exit on suspend" checkbox in iOS S/B

Users can now set the .plist property of whether their applications should exit when suspended. Added warning as this is still experimental behaviour.

Added Build Number to iOS Standalone Builder

This will add a Build Number to the plist file of the standalone builder, enabling resubmission of the same build to the app store multiple times

SVG icon support in the Extension Builder

The 'Extension Builder' now displays LiveCode Builder extensions' SVG icons, if present. You can add an SVG icon to an LCB extension by setting its "svgicon" metadata to an SVG path that could be displayed by the 'SVG Icon' widget.

When an extension provides an SVG icon, packaging the extension no longer requires you to choose bitmap icon files.

Show up to 10 nested behavior in the Project Browser

It is now possible to view up to 10 nested behaviors of an object in the PB. The behaviors are shown using oval graphics. Clicking on the graphic takes you to the script of the behavior. The tooltip of the graphic shows the long name of the behavior.

<Shift+Tab> reformats entire script

Holding down the Shift key while pressing the Tab key will reformat the entire script in the Script Editor.

Specific IDE bug fixes (9.0.0-dp-9)

18088 Allow setting multi-line tooltips from the Property Inspector
18366 Enable control+tab on MacOS
18739 Dictionary auto-search on first char freezing cursor
18881 Dictionary: sort API menu
19031 Searching the Dictionary for $
20064 Show only valid provisioning profiles on iOS Standalone Settings
20140 breakpoints better tracking of editor scrolling
20145 Move breakpoints appropriately when editing scripts
20220 Ensure fillGradient window displays its full content
20289 Improve dictionary sort ("me" was difficult to find)
20330 Show error dialog when the name of the stack contains quotes
20345 Ensure the IDE reopens a DB connection if this was previously closed by the user
20386 Add an options menu for a number of script editor preferences
20388 Layout menu buttons in script editor for platforms where they are visible
20392 Type over closing brackets if they match the next char

Specific IDE bug fixes (9.0.0-dp-8)

19897 Downgrade status of HTML5 builds from "highly experimental" to "experimental"
19961 Prevent property inspector 'set type' property editor from expanding at will
19988 Add Edit Behavior Script contextual menu item

Specific IDE bug fixes (9.0.0-dp-7)

14163 Fixed bug preventing users from adding Build Numbers to S/B's
15218 Reduce PI size to available screen space where necessary
17583 Allow increased height in inspector windows
19384 Set JAVA_HOME at startup if not set
19406 Correctly re-enable debugger when setting Script Debug Mode
19491 Allow PI to be resized to smaller than content height
19821 Set the default of the Mac S/B to 64-bit
19832 Format multi-line message box when pasting script

Specific IDE bug fixes (9.0.0-dp-6)

19339 Fix "Pending Messages" tab of the message box in Business Edition

Specific IDE bug fixes (9.0.0-dp-5)

17851 Add default script to scrollbar
18682 Ensure debugger ignores breakpoints and errors if a modal stack is presented
18931 Update widget creation docs with extension store instructions
19072 Add slider to PI for startAngle and arcAngle for oval graphics

Specific IDE bug fixes (9.0.0-dp-4)

18932 SVG icon support in the Extension Builder
18937 <Shift+Tab> reformats entire script
18956 Make sure oauth2 library is loaded correctly
18966 Remove size limitation for creating graphics

Specific IDE bug fixes (9.0.0-dp-3)

18037 Apply property defaults from metadata when testing widgets
18920 Reinstate that a single char can be selected with the mouse in ScriptEditor

Specific IDE bug fixes (9.0.0-dp-2)

18595 Clicking left of text now moves caret to the beginning of text
18631 Only use development team preferences when running from the repository
18644 Deactivate breakpoints correctly
18835 linkVisitedColor and linkHiliteColor can now be set from property inspector

Specific IDE bug fixes (9.0.0-dp-1)

13997 Fix issue creating breakpoints via the new breakpoint dialog
15830 Improve user feedback for invalid breakpoint conditions
18043 Add warning about numerical names to user guide.
18355 Bring script editor and documentation stacks to front if the stack is already open when navigating to content
18475 textFont of control does not get set when tabbing out of textFont comboBox in P.I.

LiveCode Community Plus IDE changes

Specific IDE bug fixes (9.0.0-dp-9)

20367 **Ensure common handlers are listed first**
20370 **Group completions when many variants of the same command are listed**
20387 **Handle desktopChanged correctly in autocomplete**
20393 **Add Cmd/Control+right as autocomplete shortcut**
20403 **Fix insertion of `tab` when autocomplete action performed on `tabKey`**
20407 **Use arrow key right to apply completions because tab can be ambiguous**
20409 **Fix selection when placeholder selected and user keys arrow right**

LiveCode Indy IDE changes

Autocomplete Pro

Autocomplete Pro extends autocomplete with the following features:

- Completions generated dynamically by introspecting the object being edited and its message path
- An Autocomplete Snippet Manager dialog is accessible from the script editor menubar to manage a custom set of completions.

Specific IDE bug fixes (9.0.0-dp-9)

20385 **Fix execution error when opening the snippet manager**

Dictionary additions

- **difference** (*command*) has been added to the dictionary.
- **messageDigest** (*function*) has been added to the dictionary.
- **mobileDisableNFCDispatch** (*command*) has been added to the dictionary.
- **mobileEnableNFCDispatch** (*command*) has been added to the dictionary.
- **mobileIsNFCAvailable** (*function*) has been added to the dictionary.
- **mobileIsNFCEnabled** (*function*) has been added to the dictionary.
- **msgChanged** (*message*) has been added to the dictionary.
- **nfcTagReceived** (*message*) has been added to the dictionary.
- **recordFormats** (*function*) has been added to the dictionary.
- **symmetric difference** (*command*) has been added to the dictionary.

Previous release notes

- LiveCode 8.1.6 Release Notes

- LiveCode 8.1.5 Release Notes
- LiveCode 8.1.4 Release Notes
- LiveCode 8.1.3 Release Notes
- LiveCode 8.1.2 Release Notes
- LiveCode 8.1.1 Release Notes
- LiveCode 8.1.0 Release Notes
- LiveCode 8.0.2 Release Notes
- LiveCode 8.0.1 Release Notes
- LiveCode 8.0.0 Release Notes
- LiveCode 7.1.4 Release Notes
- LiveCode 7.1.3 Release Notes
- LiveCode 7.1.2 Release Notes
- LiveCode 7.1.1 Release Notes
- LiveCode 7.1.0 Release Notes
- LiveCode 7.0.6 Release Notes
- LiveCode 7.0.4 Release Notes
- LiveCode 7.0.3 Release Notes
- LiveCode 7.0.1 Release Notes
- LiveCode 7.0.0 Release Notes
- LiveCode 6.7.9 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