LiveCode 9.0.0-dp-6 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 8.2 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

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)

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
```

Engine changes

LiveCode Builder Language (9.0.0-dp-6)

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.

LiveCode Builder Tools (9.0.0-dp-6)

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

New NFC tag feature on Android (9.0.0-dp-5)

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.

revvideograbber end-of-life on Mac OS X (9.0.0-dp-5)

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**

Script-only deploy (9.0.0-dp-5)

It is now possible to use script-only stacks in the mainstack and auxiliary stack parameters to the deploy command.

Specifying local host and port when opening a socket (9.0.0-dp-5)

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.

Message box refactor (9.0.0-dp-5)

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.

LCB modules can declare Android app permissions and features (9.0.0-dp-5)

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" />
```

Maximum text length on iOS native input fields (9.0.0-dp-4)

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
```

Throw error when changing behavior from behavior script (9.0.0-dp-4)

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.

Support for loading multi-module bytecode files (9.0.0-dp-4 - 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>".

Important: This feature is currently experimental. This means 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.

Calling JavaScript from HTML5 (9.0.0-dp-2)

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
```

Re-written LCB VM (9.0.0-dp-2)

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.

Undocumented multi-file libUrlMultipartFormAddPart removed (9.0.0-dp-2)

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.

libURLSetStatusCallback no longer requires a target object for the message (9.0.0-dp-1)

Passing an object reference as a second parameter to libURLSetStatusCallback is no longer required. If no object is passed in then the message will be sent to revLibURL itself and you can handle the message anywhere in the message path.

Platform support end-of-life (9.0.0-dp-1)

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

Field tab alignments in htmlText and styledText (9.0.0-dp-1)

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"]
```

Specific engine bug fixes (9.0.0-dp-6)

17318	**Fix format of mobileControlDo entry**

Specific engine bug fixes (9.0.0-dp-5)

11170	Disable keyboard suggestion when entering password in Android native input field.
11727	Fix arrow key not work in Android field
14238	Ensure background pattern stays aligned in long fields
16044	Corrected documentation for dateItems - Day of the week

18058	Fix keyboard not show in landscape orientation
18245	Message box refactor
18358	Make sure mobileControlGet does not return rounded values of startTime/endTime/currentTime
18403	Support hidden paragraph property in styledText arrays
18454	Allow socket to send broadcast packet on Android.
18619	Delete slash at the end of specialFolderPath("resources") to be consisten with other result of special folders.
18622	Ensure empty maps to empty lists when calling LCB library handlers
18833	Don't change name of tsNet stack during standalone build
18850	LCB modules can declare Android app permissions and features
18946	Fix browserNavigateComplete not firing when document has frames
18998	Fix go url for script-only stacks
19060	Ensure error when binding widget is caught correctly
19065	Improve error reporting for calling LCB library handlers
19116	Ensure tsNetGetStatus reports transfer status as "uploading" appropriately during POST requests
19138	Update OpenSSL to version 1.1.0d
19154	Fix widget browser stuck on handling javascript
19180	Crash when cloning an empty MCObjectPropertySet
19192	Add ios and android to "allowDatagramBroadcasts" dictionary entry
19200	Make sure printSettings are set correctly
19212	Prevent a crash when calling mobileComposeMail with just one param (tSubject)
19215	Make sure botton icons are present in standalones when building for multiple platforms
19248	When standalone builder removes the target stack from memory it now removes any stackfiles stacks that are in memory as well.
19293	Server returns 'ELF' over HTTP

Specific engine bug fixes (9.0.0-dp-4)

11039	Throw error when changing behavior from behavior script
13150	Ensure tabStops property docs describe relationship with indent properties
13696	The "volumes" function is only supported on Mac & Windows
13880	Fix formatting in scrollbarWidth property documentation
14172	Clarify that the "combine" command works in lexicographic key order
14247	Clarify insertion point location when field is focused
14363	The "startup" message is sent to the first card of the initial stack
14473	Provide a complete example for revZipAddItemWithData
14801	Correct examples in "split" command documentation
14867	The tabStops property can't be set to a boolean
15117	The "lineOffset" function can search for multiline substrings
15470	Correct "tool" property docs to be clear that it is not a stack property
15604	Update dictionary links to PCRE pattern documentation
16491	Fix bad example of using the menuName property

16511	Make examples of "borderPixel" use its main synonym
16658	Document the fact that "the environment" may be "server"
18853	Support for loading multi-module bytecode files (experimental)

Specific engine bug fixes (9.0.0-dp-2)

12196	Correct documentation for "do" command
18147	The scriptExecutionErrors property not listed in dictionary
18231	Fixed documentation formatting issues for binaryEncode and binaryDecode
18350	Fix spurious type errors for repeat variables in LCB
18495	Undocumented multi-file libUrlMultipartFormAddPart removed
18539	Don't change the defaultFolder on startup
18651	Ensure "10 garbage" is never a number
18743	Fix missing cross-references in "keys" dictionary entry
18774	Fix errors in "write to file" dictionary entry
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
14651	There is no documentation entry for "currentcard"
15865	Fixed Dictionary description for "is not among"
16211	Fix compilation errors with MacOSX SDK 10.10 and higher
18111	Make PDF user guide typography match dictionary view
18125	Fix Dictionary example for is within
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
18465	Syntax: mouseUp mouseButtonNumber
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

IDE changes

Show up to 10 nested behavior in the Project Browser (9.0.0-dp-4)

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.

SVG icon support in the Extension Builder (9.0.0-dp-4)

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.

<Shift+Tab> reformats entire script (9.0.0-dp-4)

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

Create script only stack behavior (9.0.0-dp-3)

The menu for assigning a behavior to a control has two additional options:

- Create behavior from new script only stack
- Create behavior using control script and script only stack Either option will prompt you for a stack name and a location for the script only stack. The new stack will be saved, assigned as the behavior of the control, and then added to the stackfiles property of control's stack.

Drag and drop stackfiles (9.0.0-dp-2)

You can now drag and drop stack files onto the stackFiles field in the PI.

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)

17448	Make sure messages are sent when going to stacks/cards from the Project Browser
17851	Add default script to scrollbar
18035	Make sure the gradient popup stack is displayed as expected
18485	Ensure relayering menu items don't relayer objects out of owner groups
18549	Make sure `lock cursor` works in the IDE
18682	Ensure debugger ignores breakpoints and errors if a modal stack is presented
18931	Update widget creation docs with extension store instructions
18991	Disable custom property editor when no node selected
19072	Add slider to PI for startAngle and arcAngle for oval graphics
19152	Show warning if the new stack name begins with "rev"
19153	Ensure objects can not be dragged to invisible open stacks from the tools palette
19160	Make sure the S/B respects the "iPad initial orientations" settings
19177	Update guide images for LiveCode 8
19178	Add test to ensure default folder doesn't change when loading IDE

19179	Add tests for standalone builder inclusions
19181	Ensure tutorial has location set when being skipped
19188	Make outputting debug vars from message box work in all contexts
19195	Allow vertical scrolling in "Value" field of Variable Visualizer window
19196	Ensure extension is installed before deleting files

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
18637	Fix searching in "Stack File and its stack files" from the script editor
18644	Deactivate breakpoints correctly
18835	linkVisitedColor and linkHiliteColor can now be set from property inspector
18878	Setting stackFiles in PI causes an error if you "cancel" the file dialog or select multiple files
5787	Drag and drop stackfiles

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 Builder changes

LiveCode Builder Standard Library

Java utilities

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`

Foreign function interface

- When passing a Number to one of the foreign integer types (`LCInt`, `LCUInt`, `IntSize`, `UIntSize`), an error will be thrown if the value is outside the range of the requested type.

- The `IntSize` and `UIntSize` types can hold the full 64-bit integer range, however the maximum magnitude which is supported for converting to and from Number is 2^53. An error will be thrown for any conversions outside this range.

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

- 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 Tools

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

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
```

LiveCode Builder Documentation

Style guide

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

Specific LCB bug fixes (9.0.0-dp-6)

19066	**Improve LCB 'execute script' command.**

Specific LCB bug fixes (9.0.0-dp-5)

19046	Ensure error is reported for undeclared identifiers
19059	Add access to script delimiters
19067	Ensure an error is thrown if there is no script access
19214	Increase usable range of IntSize and UIntSize types
19244	Nil pointers should bridge to nothing

Specific LCB bug fixes (9.0.0-dp-4)

18107	Do not permit namespace operator in unqualified identifiers.
18929	Update LiveCode Builder ABI version for LiveCode 9

Specific LCB bug fixes (9.0.0-dp-1)

18086	Improve and expand LCB style guide
18385	lc-run: Load multi-module bytecode assemblies.
18463	Show correct error position when source line includes tabs

LiveCode extension changes

Spinner widget

Spinner widget

A new spinner or activity indicator widget has been implemented. Spinners provide visual feedback to users use when performing an activity for an unknown duration such as processing a large amount of data or presenting a complex user interface.

Line Graph widget

Markers

- The new **markerScale** property controls the size of graph point markers.
- Any named icon from the SVG icon library can now be used as a graph point marker.

Tree View widget

Performance

- Previously when an array was expanded in the Tree View widget, all of the display calculations for were done before the next redraw. Now the keys are sorted (as before) but the display calculations are made for a maximum of 1000 rows. When more rows are needed due to scrolling, another 1000 are calculated at that point.

  This provides a near-continuous scrolling experience for arrays with large numbers of keys, and ties the expense of expanding an array to that of sorting its keys.

Segmented Control widget

Appearance and theming

- Dividers between segments are no longer drawn when the **showBorder** property is `false`

Properties

- Setting the **itemCount** now updates all other properties immediately, rather than at the next redraw
- All list-like properties now contain exactly **itemCount** items at all times
- The **itemNames** property may now include duplicated and/or empty segment names

JSON Library

JSON parser improvements

- `JsonImport()` no longer incorrectly accepts garbage at the end of a JSON file.
- `JsonImport()` no longer incorrectly accepts unescaped control characters in strings
- "null" is a valid JSON file, and `JsonImport("null")` no longer throws an error. It returns `nothing` in LCB and the empty string in LiveCode Script.
- A number by itself is a valid JSON file, and `JSONImport("25")` now returns 25, rather than throwing a syntax error.

JSON parser security fixes

- Some crafted JSON files could cause **JsonImport** to use excessive amounts of CPU time. The **JsonImport** function will now reject inputs with more than 500 levels of structure nesting.

mime script library

MIME Library

A new Multipurpose Internet Mail Extensions (MIME) library has been implemented to provide support for common data formats such as multipart email message bodies.

oauth2 script library

OAuth2 dialog library

A new library has been implemented for presenting an OAuth2 authorization dialog for any web service that supports OAuth2 Authorization Code Flow

getopt script library

Command-line option parsing support

The new **getopt** library provides support for parsing Linux-style command-line options.

Specific extension bug fixes (9.0.0-dp-6)

19317	**Improved description of attachments parameter for mimeEncodeAsMIMEEmail**

Specific extension bug fixes (9.0.0-dp-5)

19261	Clear selection when deleting selected node

Specific extension bug fixes (9.0.0-dp-3)

18908	Fix parsing of JSON files containing only a single-digit integer

Specific extension bug fixes (9.0.0-dp-2)

18693	Prevent long delays when expanding arrays with many keys
18697	Fix parsing of "lonely number" JSON files
18707	Fix possible denial of service via crafted JSON inputs
18714	Ensure all `itemNames`, `itemLabels` etc. can be set to empty
18779	Do not draw borders when `showBorder` is disabled

Dictionary additions

- **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.

Previous 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