LiveCode 9.0.0 Release Notes

Overview

LiveCode 9.0 enables access to libraries and platform APIs written in many other languages thanks to the community-funded 'Infinite LiveCode' project.

This includes a greatly improved LiveCode Builder virtual machine.

LiveCode 9.0 contains many additional improvements to support LiveCode app developers, including:

- A new "spinner" widget

- OAuth2 authentication library for use with web APIs (e.g. Facebook, Google and GitHub)

- A command argument parser library for building command-line standalones
- Updates and performance improvements for existing widgets

Known issues

- The installer will currently fail if you run it from a network share on Windows. Please copy the installer to a local disk before launching on this platform.
- The browser widget does not work on 32-bit Linux.
- 64-bit standalones for Mac OS X do not have support for audio recording.

Breaking changes

Standalone Building

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

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

As an attempt to improve this situation, the code that locks messages when closing and opening stacks for standalone builds has been removed. This means that where previously mainstacks would not receive openStack and closeStack messages during standalone build, they now do.

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

```
on closeStack
    if the mode of stack "revStandaloneProgress" > 0 then
        exit closesStack
    end if
end closeStack
```

LiveCode Builder

Exponentiation operator precedence

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

Using lc-compile tool in LiveCode 9:

```
-1^2 = -1
```

Using lc-compile tool in LiveCode 8:

```
-1^2 = 1
```

Platform support

The engine supports a variety of operating systems and versions. This section describes the platforms that we ensure the engine runs on without issue (although in some cases with reduced functionality).

Windows

LiveCode supports the following versions of Windows:

- Windows 7 (both 32-bit and 64-bit)
- Windows Server 2008
- Windows 8.x (Desktop)
- Windows 10

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

Linux

LiveCode supports the following Linux distributions, on 32-bit or 64-bit Intel/AMD or compatible processors:

- Ubuntu 14.04 and 16.04
- Fedora 23 & 24
- Debian 7 (Wheezy) and 8 (Jessie) [server]
- CentOS 7 [server]

LiveCode may also run on Linux installations which meet the following requirements:

- Required dependencies for core functionality:
  - glibc 2.13 or later
  - glib 2.0 or later
- Optional requirements for GUI functionality:
  - GTK/GDK 2.24 or later
  - Pango with Xft support
  - esd (optional, needed for audio output)
  - mplayer (optional, needed for media player functionality)
  - lcms (optional, required for color profile support in images)
  - gksu (optional, required for privilege elevation support)

Note: If the optional requirements are not present then LiveCode will still run but the specified features will be disabled.

Note: The requirements for GUI functionality are also required by Firefox and Chrome, so if your Linux distribution runs one of those, it will run LiveCode.

Note: It may be possible to compile and run LiveCode Community for Linux on other architectures but this is not officially supported.

Mac

The Mac engine supports:

- 10.9.x (Mavericks) on Intel
- 10.10.x (Yosemite) on Intel
- 10.11.x (El Capitan) on Intel
- 10.12.x (Sierra) on Intel
- 10.13.x (High 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 9.2 on MacOS 10.12 (Note: You need to upgrade to 10.12.6)
- Xcode 9.2 on MacOS 10.13

It is also possible to set other versions of Xcode, to allow testing on a wider range of iOS simulators. For instance, on MacOS 10.12 (Sierra), you can add *Xcode 6.2* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 8.2*.

We currently support deployment for the following versions of iOS:

- 8.2 [simulator]
- 9.2
- 10.2
- 11.2

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)
- 8.0 (Oreo)

To enable deployment to Android devices, you need to download the Android SDK, and then use the 'Android SDK Manager' to install:

- the latest "Android SDK Tools"
- the latest "Android SDK Platform Tools"

You also need to install the Java Development Kit (JDK). On Linux, this usually packaged as "openjdk". LiveCode requires JDK version 1.6 or later.

Once you have set the path of your Android SDK in the "Mobile Support" section of the LiveCode IDE's preferences, you can deploy your stack to Android devices.

Some users have reported successful Android Watch deployment, but it is not officially supported.

HTML5

LiveCode applications can be deployed to run in a web browser, by running the LiveCode engine in JavaScript and using modern HTML5 JavaScript APIs.

HTML5 deployment does not require any additional development tools to be installed.

LiveCode HTML5 standalone applications are currently supported for running in recent versions of Mozilla Firefox, Google Chrome or Safari. For more information, please see the "HTML5 Deployment" guide in the LiveCode IDE.

Setup

Installation

Each version of LiveCode installs can be installed to its own, separate folder. This allow multiple versions of LiveCode to be installed side-by-side. On Windows (and Linux), each version of LiveCode has its own Start Menu (or application menu) entry. On Mac OS X, each version has its own app bundle.

On Mac OS X, install LiveCode by mounting the `.dmg` file and dragging the app bundle to the `Applications` folder (or any other suitable location).

For Windows and Linux, the default installation locations when installing for "All Users" are:

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

HTML5 Native Layer

Support has been added for widgets to use HTML5 elements as native layers.

The built-in emscripten module (com.livecode.emscripten) has been added to provide useful functions for interacting with JavaScript within the browser, and to convert JavaScript objects to pointers suitable for setting as the widget's native layer.

Make the encoding property of field char chunks more useful

The 'encoding' char-level field property will now return native if all chars in the chunk can be encoded in the native encoding, and unicode otherwise.

This means that the property will now return the identical value as it did in 6.7 and before, assuming that the field text hadn't had its encoding changed by script (via the textFont ',unicode' flag).

clipboard always converts plain text to styled text

Whenever `text` was placed on the clipboard, it was first converted to LiveCode styled text and then put on the clipboard as styled text, RTF, HTML, and plain text. This introduced errors when pasting to other applications since they would prefer the HTML version which made the text appear double spaced.

Return Typeface object from MCCanvasFontGetHandle on Android

The canvas module's font handle is intended to return a pointer that can be used on the given platform to set the font of native views on that platform. In the case of android, the handle was a Skia Typeface pointer rather than a Typeface Java object.

Now the returned pointer can be wrapped using PointerToJObject and used directly as the Typeface parameter in android java ffi calls (such as setTypeface).

Note, the pointer returned by MCCanvasFontGetHandle is an unwrapped jobject local ref, so can only be expected to live while the calling handler is running. If the Typeface object is required elsewhere, it must be wrapped using PointerToJObject and put in a module variable.

Script only stacks with behavior

Script only stacks can now store a stack behavior as part of the file format. A 'with behavior' clause is added to the header of a script only stack, if it has a behavior property which references a stack.

When a script-only-stack with such a clause is loaded, the behavior is set as part of the loading.

Creating multiple 'windows' in HTML5

Basic support for multiple windows has been added to the HTML5 engine. The windows are implemented as canvas elements within the HTML5 page. This allows tooltips, dialogs, and pop-up menus to work within the HTML5 engine.

This also allows multiple stacks to be opened on the HTML5 page. Stacks other than the main stack will display in windows layered above the rest of the page. Note that these windows do not have a titlebar or standard window controls, so the means to move, resize, or close stacks will need to be provided within the stack UI.

Add optional 'kind' parameter to files/folders

The files and folders functions can now take an optional second parameter 'kind'. The kind parameter can be either empty or "detailed". If empty, the normal (short) form of the function is returned, otherwise the detailed (long) form of the function is returned.

Android AAR support

Extensions can now include android AARs on which they depend. The aars must be included in the extension's code/jvm-android folder. Currently, the supported AAR contents are:

```
- Jar files
- Resources
- Manifests
```

In particular, native code contained in AARs is not yet supported.

Android manifest merging

An android manifest merging mechanism has been added to the android standalone builder. This enables manifests to be included in extension jvm-android code folders which are then merged into the main manifest at build time.

Previously, it was possible to override the *template* manifest by providing a new template in a file called AndroidManifest.xml and including it in the Copy Files list. Since the merging mechanism is more general, enables multiple manifests and does not require users to update template manifests with new template replacement strings, this feature has been removed - instead any AndroidManifest.xml files included in the Copy Files list will be merged into the main manifest at build time.

Code Library Support

Extensions can now include compiled libraries on which they depend. The libraries must be compiled for each platform and architecture they are required on and placed folders named with a platform ID in the extension code folder. The platform ID folder names are in the form:

```
<architecture>-<platform>[-<options>]
```

See the platform ID specification for more details.

On all platforms with the exception of iOS only dynamically linked libraries are supported.

On iOS 8+ dynamically linked frameworks (`.framework`) are supported and on all versions of iOS statically linked frameworks and libraries (`.a`) are supported. Static linking is not yet supported in iOS simulator builds.

If the iOS library requires linker dependencies a text file (`.txt`) may be included to list them in the form:

```
{library | [weak-]framework} <name>
```

Additionally, on iOS the `.lcext` extension is used to identify code resources that conform to the exported symbols and sectors of externals. Specifically they have a `__deps` sector that contains the content of the dependencies file mentioned above and they export a `LibInfo` struct named `__libinfoptr_<libraryname>`. Examples of generating `.lcext` files are available in the LiveCode source repository. This is a more efficient means of inclusion as it allows the compiler to strip unused symbols.

HTML5 deployment support

The test button now works for HTML5 stacks. When editing a stack with HTML5 as a deployment option, the development menu is populated with the the list of installed web browsers, with the test button launching the built HTML5 standalone in the selected browser.

Include .jar files from extensions code folders

The standalone builder will now look for jar files in `<path to extension>/code/jvm-android` and update the `jarFiles` standalone setting accordingly when building for android. It will also look for them in `<path to extension>/code/jvm` for code inclusions on all platforms that support java.

So in order to include compile java classes in an extension and access them from LCB, simply put the .jar files into the appropriate folder (`code/jvm-android` if the jar depends on the android API, and `code/jvm` otherwise).

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.

HTML5 Callbacks - enable calling handlers in LiveCode emscripten standalones from JavaScript

Note This is still an experimental feature - details may change as development continues.

Stack setup

Each stack can be configured to expose its handlers & functions to JavaScript. This is done through a custom property of the stack - `cJavascriptHandlers`. (This will be replaced in the future by a `javascriptHandlers` property which will then be a reserved keyword). The `cJavascriptHandlers` property is a return-delimited list of handler names. The named handlers do not have to be defined at the time the stack is loaded, however calling an undefined handler from JavaScript will result in an error.

Calling from JavaScript

The standalone engine will create a `liveCode` object on the DOM `document` object. To this object will be attached the `findStackWithName` method that can be called to return a JavaScript stack object. Each stack object will have methods corresponding to the exposed handlers of that stack. For instance, a stack with the `cJavascriptHandlers` property set to :

```
performAction
setProperty
getProperty
```

will have methods named accordingly, which when executed will call the corresponding handler with the provided arguments.

JavaScript Example:

```
var myStack = document.liveCode.findStackWithName(“HTMLTest”);
var oldDocTitle = myStack.getProperty(‘documentTitle’);
myStack.setProperty(‘documentTitle’, ‘Important Document’);
myStack.performAction(‘sendEmail’);
```

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

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.

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

Specific engine bug fixes (9.0.0)

13180	**Allow GPS access from Android Browser**
20677	**Fix crash calling mobilePickPhoto with no source parameter**
20961	**Clear the menu object when moving from one menu to another**
21063	**Fixed connecting to PostgreSQL databases in standalones**
21066	**Fix unloading of tsNet via dispatch command**
21074	**revDocsFormatLibraryArrayAsJSON omits dictionary data**
21078	**Apply LCB android permissions to app they are included in**
21109	**Ensure when setting `the fullClipboardData["text"]` to only clear the clipboard if it contains styled text and do not clear private data**
21124	**Ensure folders are copied in HTML5 standalone**

Specific engine bug fixes (9.0.0-rc-1)

13935	Fix modal dialog opening behind other windows on Linux
16388	Fix setting dragdata["files"] when filename has spaces
18669	Fix rendering of rotated text on Linux
19206	clipboard always converts plain text to styled text
19536	Fix menu not being dismissed when user clicks on location greater than menu right or bottom
19580	Ensure colors round-trip through styledText correctly
19652	Refresh player on windows when mirroring property is set

19731	Remove documentation for no longer supported inks
20196	Build standalone successfully, but emit warning when extension is missing
20269	Throw parse error when dispatch ... with has empty params
20272	Fix crash when binding variables for ODBC database query
20293	Fixed bug causing crash when using chunks of the type 'the last char to -4 of "fdwbfdf"'
20482	Fix crash on Android with unsupported emojis
20565	Fix setting stack to fullscreen hides all other stacks on Linux
20582	Ensure the iOS device plist has correct values for the version of Xcode and SDKs used to build the standalone
20790	Fix auto-updater text spacing
20792	Send touch messages when android scroller `scrollingEnabled` is `false`
20811	Make the encoding property of field char chunks more useful
20814	Fix printing to pdf
20823	Ensure the iOS standalone builder looks at the correct place for the font mapping file
20839	Fix font rendering in HTML5
20856	Use sub-pixel positioning when rendering text on iOS/Mac.
20875	Return Typeface object from MCCanvasFontGetHandle on Android
20876	Fix 'could not build R.java' error when deploying to android on Windows
20884	Fixed AppStore submission error when minimum deployment target is iOS 11.0 or more
20888	Tree view widget now supports "," in the keys
20893	Ensure dropboxUpload works with default values for pAutorename and pMute params
20898	Fix crash when converting from utf16 with revDataFromQuery
20907	Ensure params passed to iphoneSetDoNotBackupFile are taken into account
20928	Make multipleHilites the main entry for multipleLines property
20946	Remove 32 bit slice from Mac externals if the standalone supports 64 bit only
20952	Fix shell on Windows Server
20953	Ensure specialFolderPath("engine") returns folder containing engine on Windows
20971	Throw parse error when unary function has no params
20976	Set the hidden of a range of lines properly when the last line is empty
20986	Ensure mobileSoundOnChannel() returns correct value on iOS
20991	Use `win32` instead of `win` in code library path to conform to platform id spec
20992	Fix crash when using several android native fields
20996	Segmented Control Widget displays the curvatures right if there is only one segment
21003	Fixed malformed Dictionary entry of mobileGetContactData
21008	Fix crash on execution error in ask dialog
21010	Use PAC file found via WPAD protocol to resolve proxy server settings for url requests
21016	Add support for local storage to Android browser
21026	Fix corruption of binary data when loading URLs in HTML5
21027	Fix android shell commands with space

Specific engine bug fixes (9.0.0-dp-11)

13825	revSetStackProfile does not work on mobile
16551	Ensure the "hilitedButtonName" returns the name of the hilited button
18010	Add optional 'kind' parameter to files/folders
18097	Geometry manager does not work on mobile
18243	Ensure horizontal two finger scrolling on Linux respects the system settings

19358	Fix text font setting in mobile engines
19543	Dictionary improperly parses hard wrapped lines where word is followed by colon
20256	Ensure iOS picker subview width scales correctly
20399	Fix missing launcher icon on Linux
20419	Fix accelRender issues on Android
20467	Fixed bug causing crash when using multiple players.
20478	Prevent crash on quit when using the commandName
20481	Fix non-displaying accented characters in HTML5 standalones
20519	Prevent android native field widget crash on ime action
20538	Ensure flushEvents("all") works on MacOS
20563	HTML 5 engine does not support fetching binary data using get URL
20592	Ensure iOS standalones are treated as unique by fingerprint scanning
20603	Ensure extensions' resource folders are available in standalones
20625	Prevent crash on startup on iOS standalone if a navbar with a non default icon is included
20627	Set default timeout in tsNet to prevent app hangs when Internet connection drops
20628	Add tsNet builds for iOS SDK 11.1
20631	Fix crash when launching android app from local notification
20633	Ensure vtab doesn't interfere with styling
20641	Added support for splash screens and icon for iPhone X
20642	Fix crash when undoing a group deletion
20644	Fix calls to struct-returning obj-c methods
20654	Fix crash when trying to write to a disconnected socket
20664	Fixed typo in installer when uninstalling previous version
20670	[Xcode 9+] Detect correctly if a simulator device is already open
20688	Fix the formatting of the `sort container` command "sortType" descriptIon
20703	revDocsParseDescriptionOfEnum continuation line parse error
20712	Ensure datagrids are re-initialised after deploy
20742	Add tsNet builds for iOS SDK 11.2
20755	Fix crash when calling iPhoneSetRemoteControlDisplay
20758	Fix performance regression in replaceText
20759	Fix crash saving images to iOS photo library
20760	Fixed documentation for tsNetSetTimeouts to specify correct units for all parameters
20763	Fix crash when deleting datagrid then undoing

Specific engine bug fixes (9.0.0-dp-10)

13992	Fixed bug causing crash on mouse enter
16709	Fix the mouse function always returning false
16713	Return correct button from answer dialog when "ok","cancel" options swapped
19909	HTML5 deployment support
19918	Add support for jar files in desktop standalones
20469	LCB Canvas: Fix alpha value of previous paint color being applied to gradient or pattern paints
20503	Fix quote key not working with Turkish keyboard layout on Mac
20507	Ensure Y is respected in 'read from socket X for Y'
20509	Fix segmented control hiliteChanged docs
20510	Fix crash on Windows when using revDatabaseTableNames()

20547	Allow user classes to to be included in android standalones

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
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
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
20326	"Ask file" dialog does not appear and eventually causes crash on Mac
20347	Fix crash when cloning groups
20357	Ensure the points list is displayed correctly, without showning -32768,-32768 for empty line
20411	Fix crash when using Obj-C FFI on iOS

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.
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
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

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.
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
19502	Make sure calling LCB library handlers produces correct error stack
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
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)

13570	Dictionary - Export Snapshot added missing platform and OS elements.
17318	Fix format of mobileControlDo entry

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
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
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
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
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
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

Fix multi-module assembly support

The extension builder now supports multi-module assemblies properly. An extension folder can contain a main module, eg main.lcb, and any number of support modules, named
-.lcb, which are then compiled together into one bytecode file.

For example, it is useful when building cross-platform extensions to put the platform-specific code in a support module, so the extension folder would contain for example

```
button.lcb
button-android.lcb
button-ios.lcb
...
```

etc

Add extension search folders preference

Any folder containing a folder containing an extension selected for loading by the user is added to the `cDeveloperExtensionsFolders` preference. There is now a new preference `cDeveloperExtensionsActiveFolders` which controls which of the folders are actually searched for extensions in the extension builder. This preference can be set using the cog at the top-right of the extension builder stack.

DataGrid 2: Edit Mode and Swipe Actions

The DataGrid has been updated to to include an edit mode and swipe actions. This only applies to DataGrids in form mode.

When in edit mode, customizable action and reorder controls will be displayed for each row, allowing for action handling and dynamic reordering.

When enabled, swipe actions allow the user to drag rows left and right as well as swiping them.

Edit Mode

A DataGrid can be put into edit mode by setting its *dgEditMode* property.

```
set the dgEditMode of group "DataGrid 1" to true
```

Only form DataGrids with fixed row height can be put into edit mode.

Action Select and Reorder Controls

By default, when in edit mode, an action select control will appear on the right hand side of each row and a reorder control on the left.

You can customize the appearance of these controls in 3 ways:

The first method is to specify your own controls using the the properties "*edit mode action select control*" and "*edit mode reorder control*".

```
set the dgProps["edit mode action select control"] of group "DataGrid 1" to group
"my action select control"
set the dgProps["edit mode reorder control"] of group "DataGrid 1" to group empty
```

Setting either of these properties to empty will result in the given control not being displayed. This way you can turn reordering off, or have only the reordering controls visible.

The second method is to handle the messages *GetEditModeActionSelectControl* and *GetEditModeReorderControl* in your row behavior script.

```
on GetEditModeActionSelectControl
  return the long id of group "my action select control" of me
end GetEditModeActionSelectControl
```

You can also use this method to prevent the reordering of certain rows. For example:

```
on GetEditModeReorderControl
   if the dgIndex of me is 5 then
      return empty
   end if
   pass GetEditModeReorderControl
end GetEditModeReorderControl
```

This will mean the row with DataGrid index 5 cannot be reordered.

The third method is to directly edit the template controls, which are hidden groups your DataGrid template stack.

Laying Out Rows

When positioning controls in your row behavior script, you can take into account any edit mode controls that have been overlaid onto the row by using the rows working rect. The working rect is passed as the second parameter to the LayoutControl message that is sent to your row behavior script.

```
on LayoutControl pControlRect, pWorkingRect
   ...
end LayoutControl
```

Action Controls

By default, when an action select control is clicked, an action control will appear on the right hand side of the corresponding row. The default action control is a delete button.

The default behavior of a user click on the action select control can be overridden by handling the message *EditModeActionSelectControlClicked* in your row behavior. The default implementation is as follows:

```
on EditModeActionSelectControlClicked pTarget
   EditModeShowActionControlForIndex the dgIndex of me
end EditModeActionSelectControlClicked
```

The target of the click is passed as a parameter. The command *EditModeShowActionControlForIndex* can be used make the action control appear for the given row.

The appearance of the action control can be customized in the same way as the action select and reorder controls: By either setting the *dgProps["edit mode action control"]* of the DataGrid, handling the message *GetActionControl* in your row behavior or editing the template control in the template stack. This allows you to potentially add multiple buttons to the action control or customize the action control for specific rows.

By default, when the action control is clicked, the given row will be deleted. This behavior can be overridden by handling the message *EditModeActionControlClicked*. For example, if you want to display a confirmation dialog, you would add the following to your row behavior script.

```
on EditModeActionControlClicked pTarget
   answer "Are you sure you want to delete?" with "Yes" and "No"
   if it is "Yes" then
      DeleteIndex the dgIndex of me
   else
      EditModeHideActionControl
   end if
end EditModeActionControlClicked
```

The target of the click is passed as a parameter to the message. Use this if you want to determine where the user clicked within the action control.

The command *EditModeHideActionControl* is used here to hide the action control and return things to the standard edit mode view.

Reordering

When the user starts a dynamic reorder, the message *EditModeReorderStarted* is sent to the DataGrid.

```
on EditModeReorderStarted pIndex, pLineNo
   ...
end EditModeReorderStarted
```

The DataGrid index of the row being reordered is passed as the first parameter. The position of the row within the DataGrid is passed as the second parameter.

When a reorder has been completed, the message *EditModeReorderCompleted* is sent to the DataGrid.

```
on EditModeReorderCompleted pIndex, pStartLineNo, pEndLineNo
   ...
end EditModeReorderCompleted
```

The DataGrid index of the row reordered is passed as the first parameter. The original position of the row within the DataGrid is passed as the second parameter. The new position of the row within the DataGrid is passed as the third parameter.

Swipe Actions

To turn on swipe actions, set the DataGrid property "*enable swipes*" to true.

```
set the dgProps["enable swipes"] of group "DataGrid 1" to true
```

When enabled, users can drag and swipe rows left and right.

By default, when a user drags a row left or right, a swipe control will be revealed. When a user swipes left or right, the right or left swipe control will be fully revealed. When a user clicks on the left or right swipe control, the row will be deleted.

The appearance of the left and right swipe controls can be customized in the same way as the edit mode controls, by either setting the DataGrid properties "*left swipe control*" or "*right swipe control*", handling the messages "*GetLeftSwipeControl*" or "GetRightSwipeControl"* or editing the template swipe control in the template stack.

For example, if you only want the right hand side swipe control to appear, you can do the following:

```
set the dgProps["left swipe control"] of group "DataGrid 1" to empty
```

The default left and right swipe actions can be overridden by handling the messages "*RowSwipedLeft*" and "*RowSwipedRight*" in your row behavior.

For example, if you want left swipes to immediately delete the row rather than fully reveal the right swipe control, you can do the following:

```
on RowSwipedLeft
   DeleteIndex the dgIndex of me
end RowSwipedLeft
```

The command *RowSwipeShowControlForIndexAndSide* can be used to fully reveal a swipe control. For example:

```
dispatch "RowSwipeShowControlForIndexAndSide" to group "DataGrid 1" with 5,
"left"
```

This will result in the left hand side swipe control being shown for the row with index 5.

Any visible swipe control can be dismissed using the command *RowSwipeHideControl*

```
dispatch "RowSwipeHideControl" to group "DataGrid 1" to group "DataGrid 1"
```

The default behavior for when a swipe control is clicked can be overridden by handling the messages "*RowLeftSwipeControlClicked*" and "*RowLeftRightControlClicked*" in your row behavior script.

For example:

```
on RowLeftRightControlClicked pTarget
   switch the name of pTarget
      case "archive"
         ...
         DeleteIndex the dgIndex of me
         break
      case "delete"
         ...
         DeleteIndex the dgIndex of me
         break
      default
         RowSwipeHideControl
         break
   end switch
end RowLeftRightControlClicked
```

The target of the click is passed as the first parameter, allowing you to determine where in the swipe control the user clicked.

Enabling and Disabling Animations

By default, all of the edit mode and swipe actions will be animated. Animations can be turned off by setting the DataGrid property "*animate actions*" to false.

```
set the dgProps["animate actions"] of group "DataGrid 1" to false
```

Create mobile scroller when reopening a Data Grid

When running on mobile the DataGrid mobile scroller was not being created if a DataGrid was closed and then opened again. For example, if you navigated away from a card with a DataGrid and then returned to the card.

Custom property inspector editors

Extensions may now include their own property inspector editors for use when editing their properties. The stack containing the editor, and the behavior must be named .editor..livecode and .editor..behavior.livecodescript respectively.

The editor stack must contain one group called "template", which is the editor. It is cloned onto the inspector stack when any property that uses it as an editor is inspected.

Please see the property inspector feature note for more details on property inspector editors.

Script Extensions

Support has been added for installing script libraries contained in livecode extension packages (.lce files).

extensionInitialize and extensionFinalize messages

`extensionInitialize` and `extensionFinalize` handlers should be implemented to govern how the script library extension is loaded in the IDE and in a standalone.

Example

```
on extensionInitialize
   if the target is not me then
      pass extensionInitialize
   end if

   insert the script of me into back

   if the environment contains "development" then
      -- Do IDE-specific initialisation
   end if
end extensionInitialize

on extensionFinalize
   if the target is not me then
      pass extensionFinalize
   end if

   remove the script of me from back
end extensionFinalize
```

extensionStartupScript message

A `extensionStartupScript` handler can be implemented to provide code which will be executed after the library is loaded in a standalone application.

Example

```
on extensionStartupScript
   if the target is not me then
      pass extensionStartupScript
   end if

   -- Initialise library for use in standalone
end extensionStartupScript
```

Metadata

Metadata for script libraries should be contained in the library level docs. The following docs elements are used for library metadata:

- Title: the display name of the library
- Version: the version number (using the semantic versioning system)
- Author: the library author

- Requires: a comma-delimited list of other extension IDs (eg widget kinds, lcb or lcs library extension ids)
- Uses: a comma-delimited list of the library's requirements when building a standalone. Currently only android permissions and hardware features are supported.

Other docs elements are used for display in the docs. Any tags and associations applied to the library in this section are applied to all elements of the library's API.

Example

```
script "com.livecode.library.networking"
/**
Title: My Networking Library

Version: 1.0.0

Author: LiveCode

Description:
This library provides a networking API.

Requires: com.livecode.library.messageauthentication

Uses: INTERNET (android permission)

Tags: networking
*/

on extensionInitialize
...
```

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)

20647	**Ensure Dictionary responds to cmd+W shortcut**
21032	**Layer tab button on iOS standalone settings below other controls**
21065	**Set loc of extension builder generated test stack correctly on first test**
21082	**Notify when attempting to install package that doesn't validate**
21090	**Ensure project browser updates when DG2 controls are added to the template stack**
21093	**Fix the display of errors thrown by the throw command in the script editor**
21095	**Ensure extensions required by the IDE load on startup**
21100	**Ensure the defaultStack does not change after opening revVariableVisualizer stack**
21116	**Table style DataGrid should also have mobile scrollers**

Specific IDE bug fixes (9.0.0-rc-1)

DG2	Fixed typo preventing EditModeReorderCompleted and EditModeReorderStarted messages from being sent to the Datagrid
8704	Ensure the autoupdater on Linux can be launched
19013	Default PI color picker to RGB unless already set to color name
19741	Remove legacy inks from property inspector
20332	Show error dialog when choosing a provisioning profile that does not support the current application identifier
20558	Allow enabling NFC support in the Android Standalone Settings
20692	Add search in scripts to `Find in` script editor contextual menu
20736	Fix size of label fields in script editor search and replace dialog
20783	Import SVG from IDE file menu
20794	Show warning when creating a Map widget on OSX 10.9 and 10.10
20799	Removed errant copy/paste from DataGrid lib code
20806	Ensure Home and End Keys are not intercepted by the PI scroller when the target is a PI

	field
20830	Add `iconGravity` property to inspector
20841	Remove duplicate handlers from IDE
20851	Fix adding `break` after `case` in switch control structure when one exists
20857	Fix widget property ordering
20862	Fix multi-module assembly support
20879	Fix error in script editor find behavior on first open
20882	Ensure PI editors can control multiple props
20894	[Start Center] Ensure creating new stack with tablet Landscape button works as expected
20901	Fix issues with users' extension guides
20909	Fix formatting of indents in SE when autocomplete is false and autoformat is true
20929	Don't overwrite existing module.lcm files with new version
20979	Escape `&` in recent files and window menus
21014	Fix installing extensions from local .lce files
21022	Ensure nested groups' scripts can be searched
21044	Prevent removal of trailing empty lines when formatting script

Specific IDE bug fixes (9.0.0-dp-11)

4010	Ensure Image/Object Library places images/objects only on user's stack
11204	Script Editor: Use same background color for both editing field and handler list
17287	Image filename property is not relativized with the PI
17819	Enable cmd+c in dictionary
19359	Ensure the debugger and remote debugger ignore breakpoints and errors if they are repeatedly triggered when execution has not been continued
19946	Add extension search folders preference
19969	Geometry Manager 'Left object' alignment issue
20431	Show ask and answer dialogs in correct location when the screen top is not 0
20475	Improve `Go to definition` for behaviors in use
20514	Add a way to launch sample extension sample stacks
20535	Check for IDE stacks correctly when toggling `Suppress Messages`
20536	Check for IDE stacks correctly when suspending development tools
20560	Improve layout of dictionary filter list
20603	Fix error when building standalone with extension with resources
20618	Make ideDocsFetchLibraryEntries public
20619	Property Inspector / Geometry: Limit Object check box not selected correctly
20621	Script Editor: Fix indention for some IF forms
20645	Create mobile scroller when reopening a Data Grid
20646	Prevent error when building a custom widget's Guide
20666	Property Inspector / Geometry: Remove All button does not work
20673	Ensure custom widget's `guide.md` is added to the Dictionary Guides
20706	Ensure Ctrl | Cmd + left arrow moves cursor to beginning of line
20713	Enable lock/unlock text from popUp for fields
20725	Show red breakpoint dot immediately when set via popUp in SE

Specific IDE bug fixes (9.0.0-dp-10)

14863	Use native scrollers for DataGrid scrollbars on mobile
20439	Scroll main property inspector group with mouse wheel and navigation keys
20483	Ensure effects popup palettes have correct height
20537	Navigate the property inspector tabs with Control+tab
20557	Ensure installation of custom widgets works, and their documentation is added to the dictionary
20561	Fix 'show documentation' for widgets

Specific IDE bug fixes (9.0.0-dp-9)

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
20220	Ensure fillGradient window displays its full content
20289	Improve dictionary sort ("me" was difficult to find)

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 extension changes

iOS Native Button widget

iOS Native Button

A new widget has been added which is a native button view on iOS. On all other platforms it presents a placeholder image with the iOS logo.

Properties

- **enabled** - The enabled state of the button

- **label** - The label to display

Messages

- **mouseUp** - sent when the button is clicked

Mac Native Button widget

macOS Native Button

A new widget has been added which is a native button view on macOS. On all other platforms it presents a placeholder image with the macOS logo.

Properties

- **enabled** - The enabled state of the button
- **label** - The label to display

Messages

- **mouseUp** - sent when the button is clicked

Mac Native Single-line Field widget

Mac Native Single-line Field Widget

A new widget has been added which is a native single-line field view on Mac. On all other platforms it presents a placeholder image. The widget is a wrapper around the Mac NSTextField class.

Properties

- **enabled** - The enabled state of the field
- **textSize** - The size of the field text
- **textFont** - The font of the field text
- **text** - The text contents of the field
- **showBorder** - Whether the field has a border or not focus
- **showFocusBorder** - Whether the field displays a blue border when focused or not

Messages

- **returnKey** - sent when the return key is pressed

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.

Android Native Field widget

Android Native Field Widget

A new widget has been added which is a native field view on Android. On all other platforms it presents a placeholder image with the Android logo. The widget is a wrapper around the Android EditText class. The implementation is based on the 'input' and 'multiline input' native controls on Android, but removes the need for mobileControlCreate / mobileControlSet syntax. The widget can be dragged from the tools palette and have properties set on it in the usual way. Moreover the widget's object is the target of its messages, rather than the object whose script created the control.

passReturnKey property

- A `passReturnKey` property has been added to allow the user to control whether the return key is passed to the field widget to add a new line or not.

Properties

- **enabled** - The enabled state of the field
- **editable** - Whether the field contents can be edited
- **fieldTextColor** - The color of the field text
- **textSize** - The size of the field text
- **multiline** - Whether the field can contain multiple lines of text
- **scrollingEnabled** - Whether the field contents can be scrolled
- **text** - The text contents of the field
- **textAlign** - The (horizontal) text alignment of the field
- **verticalTextAlign** - The (horizontal) text alignment of the field
- **autoCapitalizationType** - The auto-capitalization behavior of the field
- **autoCorrectionType** - The auto-correct behavior of the field
- **contentType** - Whether the field content is plain text or password
- **dataDetectorTypes** - The types of field data that are displayed as URLs
- **keyboardType** - The type of keyboard to display when the field has focus
- **returnKeyType** - The action associated with the return key

Non-persistent properties

- **selectedRange** - The start and length of the field text selection
- **focused** - Whether the field is focused or not

Messages

- **openField** - sent when the field gains focus
- **closeField** - sent when the field loses focus with its content having changed
- **exitField** - sent when the field loses focus with no change in content
- **returnKey** - sent when the return key is pressed

HTML5 Native Button widget

HTML5 Native Button

A new widget has been added which is a native button view on HTML5. On all other platforms it presents a placeholder label.

Properties

- **label** - The label to display

Messages

- **mouseUp** - sent when the button is clicked

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

`horizontal` property

- A `horizontal` property has been added to the segmented control which controls the orientation of the segments.

toggleHilites

- A `toggleHilites` property has been added to the segmented control. This controls the whether a currently hilited segment can be unhilited by clicking on it.

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

Android Native Button widget

Android Native Button

A new widget has been added which is a native button view on Android. On all other platforms it presents a placeholder image with the Android logo.

Properties

- **enabled** - The enabled state of the button
- **label** - The label to display
- **labelColor** - The color of the button label text
- **textSize** - The size of the button label text

Messages

- **mouseUp** - sent when the button is clicked

Time zone Library

Time zone library

A time zone library has been added for converting to and from universal time in various time zones. It uses the IANA timezone database to do the conversion.

The library contains three handlers:

- `ToUniversalTime(pSeconds, pTimeZone)` - convert from local time in the specified time zone to universal time
- `FromUniversalTime(pSeconds, pTimeZone)` - convert from universal time to local time in the specified time zone
- `TimeZones()` - list of valid time zones, one per line

Toast Notification Library

Toast Library

A toast library has been added. This provides the ability to pop up a transient, non-modal notification to the user. Currently, the library is only supported on android. The library places the following handlers in the message path:

- `mobileToast pMessage, pDuration`: Display a toast for the specified duration
- `mobileToastCancel`: Cancel the current toast

Icon SVG Library

Add icon / Set current family

- Icons can be added individually. The default family is "custom". Added icons can specify the family in the name using the form "family/name".

- Multiple icons can be added using an array (same format that `iconData()` returns). The family must be specified.

- The current icon family can now be set. This allows existing tools in the IDE to show icons from any loaded family.

New handlers:

- `iconFamilies()` - returns a list of current icon families in the library
- `iconNamesForFamily()` - returns a string list of icon names for a specified family
- `iconListForFamily()` - returns a LCB list of icon names for a specified family
- `iconDataForFamily()` - returns LCB array of icon data for a specified family
- `addIcon()` - add an icon to the iconsvg library
- `addIconsForFamily()` - add a family of icons to the iconsvg library
- `getCurrentIconFamily()` - get the family used as default (i.e. for iconPicker)
- `setCurrentIconFamily()` - set the family to be used as default (i.e. for iconPicker)
- `deleteIconFamily()` - remove a family from the library
- `iconArrayMatchingInAllFamilies()` - search for icon matches among all families

Android Audio Recorder library

Android Audio Recorder

An android audio recorder has been added. As well as starting and stopping recording, it allows the selection of various input sources, encoding types and output formats. The library places the following handlers in the message path:

- `androidRecorderStartRecording pFileName`: Start recording an audio file, using the given filename
- `androidRecorderStopRecording`: Stop recording
- `androidRecorderGetMaxAmplitude`: Returns the max amplitude of the recording since last sampled
- `androidRecorderSetRecordCompressionType pType`: Set the record compression type
- `androidRecorderSetRecordFormat pFormat`: Set the record output format
- `androidRecorderSetRecordInput pSource`: Set the record input source

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.

Widget Utilities module

Placeholder painting

Two functions have been added to facilitate drawing placeholders on the widget canvas for native widget display on other platforms:

- `placeholderIcon`: Takes an operating system parameter and returns an appropriate svg icon for a placeholder
- `paintPlaceholderImage`: Paint a placeholder for the widget on the given canvas

See the documentation for more details on the individual handlers and their syntax.

Android Utilities module

Android Utility Module

A new android utility module has been added. It currently contains two handlers:

- `StringToAndroidColor`: convert a string representing a color (optionally with alpha component) to an Android Color object.
- `ApplicationContext`: returns the current application Context object.

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 dialog library

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

Extension Package Utilities library

An extension package utilities library has been added. It contains handlers for managing LiveCode extension packages.

API

The following handlers are implemented:

- `extensionFetchSourceFromFolder pFolder, rSource, rSupportFiles, rType`: Fetch basic information about extension source from a folder
- `extensionPackageScriptExtension pSourceFolder, pTargetFolder, pRemoveSource`: Package a LiveCode Script extension
- `extensionPackage pSourceFolder, pTargetFolder, pRemoveSource`: Package an extension
- `extensionFindInFolder pFolder, pIsUserFolder, pRecursive, xDataA`: Find any

extensions in the given folder
- `extensionDepsOrder pLCCompile, pExtensionList`: Use lc-compile to order LiveCode Builder extensions by dependency.
- `extensionCheckModuleVersion pLCCompile, pModule`: Check the module bytecode is the same version as a particular lc-compile
- `extensionFetchMetadata pManifestPath`: Fetch metadata from an extension manifest
- `extensionExtractDocs pDocsParser, pOutputDir`: Extract docs from extension files
- `extensionBuildPackageAndExtractLCB pDocsParser, pTargetFolder, pCommercialExtension, pLCCompile, pLCIPath, pExtraLCIPaths`: Build, package and extract LiveCode Builder extensions
- `extensionOrderByDependency pExtensions, pRequiresA, pIncludeBuiltin`: Order extensions by dependency using requires info
- `extensionLCCompileVersion pLCCompile`: Return the bytecode version of a particular lc-compile
- `extensionCompile pLCCompile, pFile, pSupportFiles, pLCIPath, pExtraLCIPaths, pTargetFolder, pOutputFilename`: Compile a LiveCode Builder extension file
- `extensionBytecodeFilename pLCCompile, pUseVersion`: Return the default bytecode output filename
- `extensionProtectStack pProtectifyScript, pSourceFolder, pSourceFileName, pOutput, pGitHash`: Protect a livecode stack using the provided script

Command-line option parsing support

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

QR Code Generator Library

The public domain licensed QR Code Generator Library (sQuiRt) by John Craig has been added to the community repository for ease of maintenance and use.

diff library

A new library has been implemented for computing diffs between text sources and applying those diffs to text.

Dropbox API v2 Library

A derivative of Gerard McCarthy's Dropbox API v2 library for LiveCode is now available as a standalone inclusion. The changes to the library include a new command based API with asynchronous callback options for editions of LiveCode that support asynchronous HTTP POST.

Message Authentication Support

The new **Message Authentication** library provides support for computing a HMAC (Keyed-Hashing for Message Authentication) using a data using a specified key and message digest type as described in RFC 2104.

HTTP Server Library

The HTTP Server library can be used to recieve and respond to HTTP requests in your application.

Specific extension bug fixes (9.0.0)

21067	**Ensure extension folder searching recursion doesn't error**
21069	**Display button name as label if label is empty**
21085	**Ensure browserUnhandledLoadRequest is sent for unknown protocols**
21096	**Include all of extension guide folder in packages**
21099	**Fix crash when closing browser widget**
21105	**Include interface file in package**
21107	**Fix some `target=_blank` links not navigating**

Specific extension bug fixes (9.0.0-rc-1)

19427	Fix CEF render process not closing when widget closes
19696	Dropbox library was only returning the error code, not the description
20200	On macOS, Windows and Linux navigate to `target=_blank` links to match mobile behavior
20276	Use selected font for android native field
20276	Use selected font for android native button
20575	Add Cancel button to OAuth2 dialog

Specific extension bug fixes (9.0.0-dp-11)

19851	Implement object repository to store object references that might mutate
20515	Prevent segmented control fill from ending one pixel to the left/bottom of segment.
20556	dropboxGetCurrentAccount changed to overcome a quirk in Dropbox's v2 API that was causing an error to be returned
20663	Make corner radius of segmented widget a user settable property
20710	Fix ability to focus on Android Native Field widget
20714	Show soft keyboard when Android native field widget receives focus

Specific extension bug fixes (9.0.0-dp-10)

20461	Ensure togglehilites and multipleHilites work together
20509	Segmented control posts hiliteChanged when created
20516	Switching orientation did not properly redraw widget.
20518	Set correct keyboard input type for passwords
20534	Ensure segmented widget created in LC 9.x opens in LC 8.1.x too

Specific extension bug fixes (9.0.0-dp-9)

20081	Keep setting of enabled property in sync
20115	Fix crash when focusing field widget
20216	Fix incorrect item delimiter in request callback example
20238	Ensure request limit is an integer
20306	Ensure `dropboxGetMetadata` param `pIncludeMediaInfo` defaults to `false` and fix copy/paste error in docs

Specific extension bug fixes (9.0.0-dp-7)

18698	Ensure only well-formed JSON numbers are accepted

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-3)

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

Specific extension bug fixes (9.0.0-dp-2)

7159	JSON parser improvements
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

LiveCode Community Plus engine changes

Specific engine bug fixes (9.0.0-dp-11)

19793	Created additional server engine version for Business license holders

LiveCode Community Plus IDE changes

Autocomplete Shortcuts

The `F1` key is now used for presenting autocomplete when the user has the live autocomplete preference off. If the completions pane is already presented the `F1` key will launch the dictionary if the chosen completion is documented.

Specific IDE bug fixes (9.0.0-dp-11)

20407	Revert to using tab to apply completions
20596	Parse messages with optional parameters correctly
20599	Use `F1` as the autocomplete shortcut due to the conflict with `Ctrl+Shift+right` selecting the word to the right

LiveCode Indy engine changes

Quality Presets

The read only `qualityPresets` and read-write `qualityPreset` properties have been implemented for the camera control on Windows. While on Mac and iOS the preset name is defined by the AFVoundation framework on Windows they are are in the form:

```
<width>x<height>|<frame duration in 100 nanosecond units>|<bit rate in bits per
second>
```

Exposure mode

The windows camera control now supports listing the `exposureModes` the device supports and getting and setting the `exposureMode` of the device.

Focus mode

The windows camera control now supports listing the `focusModes` the device supports and getting and setting the `focusMode` of the device.

White Balance mode

The windows camera control now supports listing the `whiteBalanceModes` the device supports and getting and setting the `whiteBalanceMode` of the device.

Allow cameraControl to record without video, or without audio.

The videoDevice & audioDevice cameraControl properties now accept the empty string in order to disable recording from those input devices.

LiveCode Indy IDE changes

Specific IDE bug fixes (9.0.0-dp-11)

20591	Present warning before deleting a snippet in the snippet manager

LiveCode Indy extension changes

Map widget

Native Map Widget

A new widget has been added which is a native map view on Android, iOS and 64-bit Mac. On all unsupported platforms and architectures it presents a placeholder image with a map pin icon. The widget is a wrapper around the GoogleMap class on Android and the MKMapView class on iOS / Mac.

Properties

- **centerCoordinates** - the coordinates of the current center of the map view in degrees.
- **span** - The extent of the map displayed by the widget in degrees.
- **region** - The region displayed by the map - the centerCoordinates followed by the span.
- **animateMovement** - Whether the movement of the map view is animated or not when setting properties.
- **showsUserLocation** - Whether to show the user's current location on the map.
- **scrollEnabled** - Whether the map view can be scrolled by user actions.
- **zoomEnabled** - Whether the map view zoom can be changed by user actions.
- **markers** - Marker points on the map. These must have coordinates and a title, and can optionally have a subtitle
- **polylines** - Polylines drawn on the map. These must have a list of coordinates and can optionally have a color and width.

Messages

- **markerSelected** - sent when a marker point is selected
- **regionChanged** - sent when the map view region is changed

Signature widget

Signature Widget

A signature widget has been implemented for drawing signatures. Line width is governed by drawing speed and the drawn line is smoothed with a bezier curve.

Secure Key Storage library

Secure Key Storage Library

A new library has been implemented for the management of secure key storage using the keychain on iOS, Mac and Android.

Amazon Web Services (AWS) Library

The new **Amazon Web Services** library includes support for Amazon Simple Storage Service (S3). You can use S3 to provide cloud storage for your app. Read more here

Specific extension bug fixes (9.0.0-rc-1)

20797	Fix incorrect uses of `centerCoordinate` instead of `centerCoordinates`
20815	Add `borderWidth` and `borderColor` properties to signature widget
20861	Improve documentation of secure key storage library
20883	API key standalone setting is required on android

Specific extension bug fixes (9.0.0-dp-9)

19979	Include documentation in IDE

LiveCode Business extension changes

Script Profiler

Use the new script profiler to identify the most costly lines of code in a stack. Open a stack and choose `Start Profiling Script` from the `Development` menu to profile the topStack. When each line of a script in the stack is executed the execution time will be recorded. Choose `Stop Profiling Script...` from the `Development` menu to stop profiling and present a report.

Specific extension bug fixes (9.0.0-dp-11)

19357	Start the remote debugger session without user interaction and indicate with a green arrow on the `Test` button
20204	Add `Save` button to script profiler
20532	Ensure remote debugger is activated even if no stacks have breakpoints
20584	Ensure that the remote debugger does not load in HTML5 due to lack of socket support
20669	Filter out 0.0.0.0 from network interfaces to attempt to connect to

Specific extension bug fixes (9.0.0-dp-10)

20487	Use the `Script Debug Mode` preference to govern the inclusion of the remote debugger when building via the `Test` button
20533	Restrict the remote debugger to only handle debugger APIs when a script is being debugged remotely

Specific extension bug fixes (9.0.0-dp-7)

19356	Improve handling of user answering no to remote debugger session
19412	Ensure the script editor is returned to edit mode before clearing the remote debugger object cache
19730	Ensure Android internet permissions when including the remote debugger

LiveCode builder changes

LiveCode Builder Standard Library

Mathematical Operator Precedence

- The exponentiation operator `^` now has higher precedence than unary minus, meaning for example that -1^2 evaluates to -1 rather than 1, as it does in LiveCode Script.

System error information

- Two new expressions have been added for accessing platform-specific system error status:
    - `the system error code` evaluates to the current numerical system error code
    - `the system error message` evaluates to a string describing the current system error
- The new `reset system error` statement clears the current system error.

Foreign function interface

- Natural integer types NaturalSInt and NaturalUInt have been added to the foreign module. These map to 32-bit or 64-bit integers, depending on the bitness of the processor.
- A natural float type NaturalFloat has been added to the foreign module. This maps to float on 32-bit processors and double on 64-bit processors.
- 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 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 Host Library

Widget library

A new expression `my pixel scale` has been implemented. Use the widget's pixel scale to calculate the size of an image to draw. For example, when drawing an image to `my bounds` create an image sized using `my width * my pixel scale, my height * my pixel scale` otherwise the image will be stretched to match the pixel scale. The pixel scale is a per-window/screen property so may change if the user moves a window to a new screen. New syntax has been added to enable property triggers to be fired directly from within a widget: trigger all.

At the moment the only property trigger that exists is the ‘property listening’ mechanism available in the IDE. You should use ‘trigger all’ after a widget property has changed internally (for example because something in a native layer has changed) so that the IDE can be notified that the property has changed.

In the future single property variants and other types of trigger may be introduced.

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 Language

Objective-C dynamic instance method binding

It is now possible to bind to instance methods dynamically, by leaving empty the name of the class in the binding string. In particular this enables calling protocol-defined instance methods on class instances, which was not previously possible.

For example, to directly call the `NSAlertDelegate` method `alertShowHelp:` on an `NSAlert` instance:

```
-- bind to delegate protocol method
foreign handler NSAlertDelegateShowHelp(in pTarget as ObjcId, in pAlert as
ObjcId) binds to "objc:.-alertShowHelp:"
...
-- call alertShowHelp on an NSAlert, passing the alert itself as the
-- first parameter.
NSAlertDelegateShowHelp(tAlert, tAlert)
```

Objective-C delegate support

Handlers `CreateObjcDelegate` and `CreateObjcDelegateWithContext` have been added to the objc module which allow the creation of custom delegate objects with LCB implementations of protocol methods.

In order to create a delegate to handle a particular protocol method, pass in the protocol name as the first argument and the mapping from method names to LCB handlers as the second argument. For example, to create a selectionChanged message for an `NSTextView`, we need to create a handler

```
private handler DidChangeSelection(in pNotification as ObjcObject) returns
nothing
    post "selectionChanged"
end handler
```

and create a `NSTextViewDelegate`:

```
variable tDelegate as optional ObjcObject
put CreateObjcDelegate( \
    "NSTextViewDelegate", \
    {"textViewDidChangeSelection:": DidChangeSelection}, \
    ) into tDelegate
if tDelegate is not nothing then
    put tDelegate into mTextViewDelegate
end if
```

Optionally, a context parameter can be passed in at delegate creation time:

```
put CreateObjcDelegateWithContext( \
    "NSTextViewDelegate", \
    {"textViewDidChangeSelection:": DidChangeSelectionContext}, \
    tContext) into tDelegate

if tDelegate is not nothing then
    put tDelegate into mTextViewDelegate
end if
```

In this case the context variable will be passed as first argument of the corresponding LCB callback:

```
private handler DidChangeSelectionContext(in pContext, in pNotification as
ObjcObject) returns nothing
    post "selectionChanged" with [pContext]
end handler
```

Some protocols consist of purely optional methods. In this case the information about the protocol's methods are not available from the
objective-c runtime API. For this eventuality there are also handlers `CreateObjcInformalDelegate` and `CreateObjcInformalDelegateWithContext`.

These handlers take a list of foreign handlers as their first argument instead of a protocol name. The foreign handlers' information is used to resolve incoming selectors so that the desired LCB callback is called. For example the `NSSoundDelegate` protocol has only one method, and it is optional,

```
- (void)sound:(NSSound *)sound didFinishPlaying:(BOOL)aBool;
```

So in order to create an `NSSoundDelegate`, we need to create a list of foreign handlers, in this case just the following:

```
foreign handler NSSoundDidFinishPlaying(in pSound as ObjcId, in pDidFinish as
CSChar) binds to "objc:.-sound:didFinishPlaying:"
```

and create the informal delegate

```
handler DidSoundFinish(in pSound as ObjcId, in pDidFinish as Boolean) returns
nothing
    if pDidFinish then
        post "soundFinished"
    end if
end handler

foreign handler Objc_SetSoundDelegate(in pSound as ObjcId, in pDelegate as
ObjcId) returns nothing \
    binds to "objc:NSSound.-setDelegate:"
...

variable tDelegate as optional ObjcObject
put CreateObjcInformalDelegate( \
    [NSSoundDidFinishPlaying], \
    {"textViewDidChangeSelection:": DidChangeSelection}) \
    into tDelegate
end if
if tDelegate is not nothing then
    put tDelegate into mSoundDelegate
    Objc_SetSoundDelegate(tSound, tDelegate)
end if
```

> *Note:* Delegate properties are usually 'assigned' rather than 'retained', so it is necessary to store them in module variables until they are no longer needed. Generally the pattern required is as follows:

```
handler OnOpen()
    -- Create native view and set native layer
    -- Set native view delegate property
    -- Store view and delegate in module vars
end handler

handler OnClose()
    -- Set native view delegate property to nothing
    -- Put nothing into view and delegate module vars
    -- Set native layer to nothing
end handler
```

`the caller` syntax

`the caller` syntax has been added to the engine module. It returns the script object which called a the handler at the beginning of the current chain of LiveCode Builder handler execution.

Library post and send syntax

Syntax for post and send with no target has been moved from the widget module to the engine module, so that it can be used by library modules. In this case the message is dispatched to the current card of the default stack, as a 'global notification' would be

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.

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.

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 builder bug fixes (9.0.0-rc-1)

19760	Clarify forms of log syntax in documentation
20430	Exponentiation should have higher precedence than unary minus
20801	Fix issues with list -> NSArray and array -> NSDictionary conversion
20874	Take access type into account when importing assembly support module definitions
20874	Take access type into account when importing assembly support module definitions
20886	Fix crash when class is omitted in obj-c binding string
20887	Allow ObjcAutoreleasedId params in Obj-C delegate callback handlers

Specific builder bug fixes (9.0.0-dp-11)

20705	Add 'the architecture' syntax
20733	Allow binding to dynamic obj-c property messages

Specific builder bug fixes (9.0.0-dp-10)

20527	Parse complex java binding strings correctly
20527	Parse complex java binding strings correctly

Specific builder bug fixes (9.0.0-dp-9)

19973	Allow bind to interface callbacks with non-void return
20089	Ensure obj-c action proxy objects are deallocated correctly
20325	Add support for foreign aggregates
20325	Add support for foreign aggregates

Specific builder bug fixes (9.0.0-dp-8)

18899	Fix load image from resource file
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
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 builder bug fixes (9.0.0-dp-7)

14861	Add "reverse _" syntax for sequence types
19486	Failure to bind foreign handlers should provide meaningful error
19523	Make sure all created registers are destroyed
19779	fix bytecode generation for 'set my native layer' syntax

Specific builder bug fixes (9.0.0-dp-6)

19066	Improve LCB 'execute script' command.

Specific builder bug fixes (9.0.0-dp-5)

19046	Ensure error is reported for undeclared identifiers
19059	Add access to script delimiters

Specific builder 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 builder 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

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