LiveCode 10.0.0-dp-4 Release Notes

- Previous release notes

Overview

This document describes all the changes that have been made for LiveCode 10.0.0-dp-4, including bug fixes and new syntax.

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
- Windows 11

Note: On 64-bit Windows installations, LiveCode can run either as a 32-bit application through the WoW layer or as a native 64-bit Windows application, depending on the installer that is chosen.

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.

macOS

The macOS engine supports Intel architecture machines running the following operating system versions:

- 10.9.x (Mavericks)
- 10.10.x (Yosemite)
- 10.11.x (El Capitan)
- 10.12.x (Sierra)
- 10.13.x (High Sierra)
- 10.14.x (Mojave)
- 10.15.x (Catalina)
- 11.x (Big Sur)
- 12.x (Monterey)

The macOS engine supports Apple architecture machines running the following operating system versions:

- 11.x (Big Sur)
- 12.x (Monterey)

Note: Apple architecture support is currently experimental. To run the IDE using Apple architecture (on supported machines), you must toggle the `Open using Rosetta` option to off in the `LiveCode.app` bundle's `Get Info` pane in Finder. To build a standalone with native Apple architecture support you must explicitly choose the `macOS Apple` option in standalone settings.

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 10.1 on MacOS 10.13 (Note: You need to upgrade to 10.13.4)
- Xcode 11.3 on MacOS 10.14 (Note: You need to upgrade to 10.14.4)
- Xcode 12.4 on MacOS 10.15 and above (Note: You need to upgrade to 10.15.4)
- Xcode 13.2 on MacOS 11 and above (Note: You need to upgrade to 11.3+)

It is also possible to set other versions of Xcode, to allow testing on a wider range of iOS

simulators. For instance, on MacOS 10.14 (High Sierra), you can add *Xcode 10.1* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 12.1*.

We currently support building against the following versions of the iOS SDK:

- 12.1 (included in Xcode 10.1)
- 13.2 (included in Xcode 11.3)
- 14.4 (included in Xcode 12.4)
- 15.2 (included in Xcode 13.2)

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 x86, x86-64, ARM and ARM64 processors. It will run on the following versions of Android:

- 5.0-5.1 (Lollipop)
- 6.0 (Marshmallow)
- 7.x (Nougat)
- 8.x (Oreo)
- 9.0 (Pie)
- 10.0 (Q)
- 11.0 (R)

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.

Web

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

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

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

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 -ui [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.
	The file to which to log installation actions. If not specified, no log is

Option	Description
`-log LOGFILE`	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 -ui [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 -ui
```

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

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
```

New Features

Android

Accelerated Rendering

An OpenGL ES 3 implementation of the accelerated rendering backend has been added. This makes it possible for future versions of LiveCode to leverage the new technology to improve accelerated rendering performance. It also enables accelerated rendering to be used with Android emulators.

Barcode Scanner

Android barcode snapshot clipping

By default, if a snaphot is taken by the Android barcode scanner widget, the resulting image is clipped to the actual rect of the barcode. A new property `clipSnapshots` has been added which can be set to false to prevent this behavior, so that the resulting snapshot is the full image.

Browser

Opaque property

By default, browsers add a solid background to their content, even if that content does not specify

a background via CSS styling. This means that they completely cover any content underneath them.

To control this behavior, the `opaque` property has been added to the browser widget.

When the property is set to true (the default), this default background will be drawn.

When the property is set to false, this default background will not be drawn and the browser content will be composited with the content underneath.

Note: Due to limitations of the browser frameworks used, setting the `opaque` property to false is only supported on macOS, iOS, Android and Web. It has no effect on Windows and Linux.

Chart

There is a preview of a new widget for displaying customizable charts now available on the tools palette.

The widget is based on the highly popular chart.js, and allows easy addition of high-quality, animated charts to any stack.

There are a wide range of properties which can be used to control the type, look and content of the chart displayed - many of which are controllable by changing options on different panes of the property inspector while a chart widget is selected.

The data can be set by using either the **csvData** or **tsvData** properties (using comma-delimited or tab-delimited data respectively).

The type of chart being displayed is controlled using the **chartType** property with the options currently being `line`, `bar`, `radar`, `donut`, `pie`, `polar`, `bubble` or `scatter`.

If the chart type displays points, then the **pointStyle** and **pointRadius** properties can be used to control the shape and size of them.

All the colors used by the chart are customizable using a range of properties, such as **chartBackgroundColor**, **titleColor** and **gridLineColors**. In particular, the colors of things such as the sectors in pie charts or the bars in bar charts can be controlled using the **backgroundDataColors**.

Similarly, there are many properties allowing control over the text displayed on the chart, including the ability to configure the font, size and content of the title, subtitle, scales and legends.

For full details of all properties available, please look up the chart widget in the dictionary.

Note: The widget is still under intense development and as such there is a high chance that property names, features, and internal format may change in incompatible ways in subsequent DPs.

iOS

Accelerated Rendering

An OpenGL ES 3 implementation of the accelerated rendering backend has been added. This makes it possible for future versions of LiveCode to leverage the new technology to improve accelerated rendering performance.

LiveCode Builder

Starting Android activities and receiving results

A new Android-specific utility handler `StartActivityForResult` has been added which starts an activity defined by a specific Intent and calls a provided callback when the activity returns with any results it generates.

For more details see the entry for `StartActivityForResult` in the android-utils section of the LiveCode Builder dictionary.

Canvas current transform

New syntax has been added to enable getting the current transform of a canvas.

The canvas `device transform` property returns the affine transform from logical units to backing device pixels.

Text encoding and decoding

New syntax has been added to allow encoding strings to, and from, a variety of different text encodings.

The `encode using <text-encoding>` statement encodes a string to binary data, and the `decode using <text-encoding>` statement does the converse. In both cases, how characters map to bytes is determined by the specified `text-encoding`:

- `ascii`: the 7-bit ASCII character set
- `native`: the native character set of the platform (`MacRoman` on macOS and iOS, `Windows-1252` on Windows, and `ISO8859-1` on Android, Linux and Web).
- `utf8` : the utf-8 encoding
- `utf16`: the utf-16 encoding in the byte order of the platform
- `utf16le`, `utf16be` : the utf-16 encoding in little-endian and big-endian byte order respectively
- `utf32`: the utf-32 encoding in the byte order of the platform
- `utf32le`, `utf32be` : the utf-32 encoding in little-endian and big-endian byte order respectively

In cases where characters cannot be encoded into the specified text encoding, the character is encoded as `?`.

Example:

```
    encode "lorem ipsum" using utf8
    decode the result using utf8 into tFooString
```

Conversion between JObjectArray and List

Syntax for converting between a `JObjectArray` and a List of `JObject`s has been added to the Java utility library.

- The alias `JObjectArray` has been added to aid readability
- `ListToJObjectArray` - converts a List of JObjects to a JObjectArray
- `ListFromJObjectArray` - converts a JObjectArray to a List of JObjects

Custom property set access

New syntax has been added to allow getting and setting properties in specific custom property sets.

Example :

```
get property "uProp" of set "cSet" of my script object
set property "uProp" of set "cSet" of my script object to 0
```

Base64 encoding and decoding

New syntax has been added to allow encoding to, and decoding from, base64.

The `encode using base64` statement encodes binary data to a string, and the `decode using base64` statement does the converse.

The format of the base64 encoding is that as defined by RFC2045.

Example:

```
    encode the contents of file "foo.bin" using base64
    decode the result using base64 into tFooBinData
```

Canvas effect layer bounds

New syntax has been added to the Canvas library to allow specifying the drawing boundary when beginning a new layer for an effect. This should be a rectangle signifying the bounds of any drawing performed on the layer.

This information allows the canvas library to limit the amount of memory allocated to the new layer by restricting the size of its backing buffer to only that area required to correctly render effects based on the given drawing bounds.

Example:

```
public handler OnPaint
    // initialize effect
    variable tProps as Array

    put color [0,0,0,0.5] into tProps["color"]
    put "source over" into tProps["blend mode"]
    put 0 into tProps["spread"]
    put 3 into tProps["size"]
    put 25 into tProps["distance"]
    put 45 into tProps["angle"]

    variable tShadow as Effect
    put outer shadow effect with properties tProps into tShadow

    // initialize shape bounds
    variable tBounds as Rectangle
    put rectangle [25, 25, 75, 75] into tBounds

    // draw rounded rectangle with shadow
    begin layer with tShadow for tBounds on this canvas
    fill rounded rectangle path of tBounds on this canvas
    end layer on this canvas
end handler
```

Wrapping handlers as JS functions

The Web-specific utility function `HandlerAsJSFunction` has been updated to allow any handler to be wrapped as a JS function, instead of just those of type `JSEventHandler`.

Image validity

New syntax has been added to allow querying an image to determine if it contains valid data.

The image `is valid` operator returns a boolean value indicating whether or not its data is valid.

LiveCode Script

Aspect-ratio preserving iconGravity options

The `iconGravity` of a button can now be set to `resizeAspect` or `resizeAspectFill`. Both of these options cause the icon to be scaled whilst maintaining aspect ratio.

If `resizeAspect` is used then the icon is scaled as much as possible so that no part of it is clipped by the button's bounds.

If `resizeAspectFill` is used then the icon is scaled to fill the button's bounds, potentially causing parts of the image to be clipped.

Sequence literals

New syntax has been added to allow sequence-style arrays to be expressed as literal values, instead of having to construct them explicitly using the `put` command.

A sequence literal is delimited by `[` and `]`, with a comma-separated list of element expressions with optional trailing comma. e.g.

```
get [] -- empty sequence
get [ 1 ] -- sequence with one element
get [ 1, 2, ] -- sequence with two elements
```

The result of a sequence literal being evaluated is an array with integer keys from one up to the number of element expressions specified. Element expressions are evaluated from left to right.

A sequence literal expression is equivalent to a sequence of `put` statements. For example:

```
get [ 1, sin(0.5), cos(0.5), 1 ]
```

Produces the same result as the sequence of commands:

```
put 1 into it[1]
put sin(0.5) into it[2]
put cos(0.5) into it[3]
put 1 into it[4]
```

There is no restriction on the kind of expression which can be specified for each element.

Dictionary literals

New syntax has been added to allow dictionary-style arrays to be expressed as literal values, instead of having to construct them explicitly using the `put` command.

A dictionary literal is delimited by `{` and `}`, with a comma-separated list of key-value pairs with an optional trailing comma. Keys must be either string or integer literals. e.g.

```
get {} -- empty array
get { "foo": 1 } -- array with one entry
get { 1: "foo" } -- array with one entry
get { "foo": 1, 1: "foo", } -- array with two entries
```

The result of a dictionary literal being evaluated is an array with each specified key mapping to the

result of evaluating its corresponding value expression. Value expressions are evaluated from left to right and building the array will take into account the current setting of `the caseSensitive` local property.

A dictionary literal expression is equivalent to a sequence of `put` statements. For example:

```
get { "a": 1, "b": cos(0.5), "c": sin(0.5), "d": 1 }
```

Produces the same result as the sequence of commands:

```
put 1 into it["a"]
put cos(0.5) into it["b"]
put sin(0.5) into it["c"]
put 1 into it["d"]
```

If a the key-value list contains duplicate keys, then the result of evaluating the last value expression for a given key will be the one which is used in the final dictionary array. For example:

```
get { "a": 1, "a": 2 }
```

Will result in a dictionary array with a single key "a", with value 2.

If `the caseSensitive` property is false, and two keys are the same up to case then the value of the key will be the last evaluated value for it, as above, but the key itself will be the first. For example:

```
set the caseSensitive to false
get { "a": 1, "A": 2 }
```

Will result in a dictionary array with single key "a", with value 2. This behavior is a side-effect of the equivalence of {} to a sequence of put statements in the same order as the key-value pairs.

There is no restriction on the kind of expression which can be specified for each value.

New nothing constant

A new constant `nothing` has been added which exposes the special value the engine places in variables which have yet to be explicitly given a value.

The `nothing` value can be tested for by using the `is strictly nothing` operator.

Example:

```
local tUninitialized, tInitialized
put tUninitialized is strictly nothing -- true
put empty into tInitialized
put tInitialized is strictly nothing -- false
put nothing into tInitialized
put tInitialized is strictly nothing -- true
```

The `nothing` value behaves like `empty` when in string context, and like zero when used in number context.

Example:

```
put nothing is 0 -- true
put nothing is empty -- true
```

Note: In general it is advised to avoid the use of `nothing` except when working with external data which requires the distinction it provides from `empty`.

Variadic handler indicator

It is now possible to indicate explicitly that a handler is `variadic`, i.e. that it takes more arguments than it has explicitly declared parameters. This can be done by using the ellipsis token (`...`) as the final parameter in the handler's signature.

For example:

```
    command MyVariadicCommand pFirst, pSecond, ...
    end MyVariadicCommand
```

Use of the ellipsis token in this fashion currently has no effect either when parsing or executing a script and is there to help indicate how a handler should be called.

macOS

Apple architecture support

Native support for the arm64-based Apple architecture has been added to the macOS IDE, standalone, and server engine.

The IDE and server engine are now shipped as universal binaries with both an Intel and Apple architecture slice.

All externals shipped with LiveCode, except for XPDF and the dboracle revdb driver, also now support Apple architecture.

You can determine which architecture a running macOS engine is using with `the processor`

function in LiveCode Script, or the `the architecture` syntax in LiveCode Builder. If running as Intel architecture, these will return `x86_64`; and if running as Apple architecture, these will return `arm64`.

Note: Apple architecture support is currently experimental. To run the IDE using Apple architecture (on supported machines), you must toggle the `Open using Rosetta` option in the `LiveCode.app` bundle's `Get Info` pane in Finder.

Note: As the server engine is a bare executable, it will run using the native architecture of the host machine. If this is of concern, either use the `arch` shell command to force it run using Intel architecture, or the `lipo` shell command to remove the Apple architecture slice.

Stack Editor

Dynamic guidelines

An option to turn on dynamic guidelines in pointer tool mode when controls are moved and resized has been added to the `View` menu - simply select the `Guidelines` item toggle them on and off.

When on, and an object is being moved or resized, indicator lines will be displayed when its sides or center line up with other objects' sides or centers and the object will naturally snap to those positions in preference to others. The size of the object will be shown continually, and distances from other controls will be shown next to the relevant indicator lines when they appear.

When on, and with an object selected (but not being moved or resized), holding down the `Option` key on macOS, or the `Alt` key on Windows or Linux, will cause the distance between the selected object and control under the mouse pointer to be displayed. If there is no control under the mouse the distances between the selected object and edges of the card will be displayed instead.

Standalone Builder

Apple architecture support

Support for building macOS standalones which have native Apple architecture support has been added to the standalone builder.

You can now select whether you want to build a macOS standalone with an Intel slice, an Apple slice, or both on the macOS pane of standalone settings.

Note: Apple architecture support is currently experimental. To build a standalone with native Apple architecture support you must explicitly choose the `macOS Apple` option in standalone settings.

Web

Map

The map widget has now been implemented for Web, allowing it to be used in apps deployed to that platform. All properties, events, and syntax are implemented.

In order to use the map widget on web, a Google Maps JavaScript API key is required, and this must be set in the configuration options in the Standalone Builder for the Map Widget inclusion.

Custom HTML page

By default, when testing or building a standalone for Web, the standalone builder will generate a fixed HTML page which embeds the app.

If you want to use your own custom HTML page you can now do so by adding a file named `standalone.html` to the `Copy Files` pane in the standalone builder.

Browser

The browser widget has now been implemented for Web, allowing it to be used in apps deployed to that platform.

The following properties are currently implemented:

- allowUserInteraction
- htmlText
- url
- userAgent
- javaScriptHandlers

The following events are currently implemented:

- browserNavigateBegin
- browserNavigateComplete
- browserDocumentLoadBegin
- browserDocumentLoadComplete

The following engine syntax is currently implemented:

- go forward in widget "myBrowser"
- go back in widget "myBrowser"
- do "myJavaScriptFunction(param1,param2);" in widget "myBrowser"

The implementation uses an HTML5 IFrame which is embedded on the hosting web page and as such is restricted by the same security policies (CORS) as that page. In particular, if the URL of the browser widget comes from a different domain to that of the hosting web page then it is likely that the following features will not work:

- executing JavaScript in the browser
- setting JavaScript handlers
- getting and setting the userAgent property
- going back and forward

For more details about such security policies see https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

WebAssembly

The LiveCode engine is now deployed to HTML5 in WebAssembly format. WebAssembly is supported in most major web browsers.

The WebAssembly engine is smaller, and substantially faster than the previous asm.js version.

In addition, moving to this new architecture means that the `wait` command now works - along with any engine functionality which internally uses wait-like functionality.

Breaking Changes

Browser

Handling unrecognised URL schemes

The browser will no longer automatically handle non-standard url schemes on Android, bringing it in line with all other platform browser implementations.

Instead, a `browserUnhandledLoadRequest` message will be sent allowing an app to provide custom behavior in this case.

To retain the previous behavior, use `launch url` in response to this message:

```
on browserUnhandledLoadRequest pUrl
   launch url pUrl
end browserUnhandledLoadRequest
```

LiveCode Builder

Web architecture identifier changed

The way the web platform's architecture is identified has changed to better reflect the use of wasm. rather than pure JavaScript previously.

The return value of `the architecture` is now `wasm` instead of `js`.

Example: "' if the architecture is "wasm" then -- previously "js" put true into sIsWasm else put false into sIsWasm end if "'

Any existing code which branches to detect the web architecture using this syntax will need to be updated to take this change into account.

LiveCode Script

New reference token

The character `@` is now reserved as the 'reference' token.

Previously, `@` was allowed to appear at the start of constant, variable and handler names. For example `@tFoo` was a valid name.

Such usage will now cause an error so any scripts using such names will need to be updated to accomodate this change.

Note: Using `@` after the first character of a constant, variable or handler name is still supported but is deprecated and should be avoided in new scripts.

Web platform identifier changed

The way the web platform is identified has been changed.

The return value of `the platform` (and its alternative form `platform()`) is now `web` instead of `html5`.

Example:

```
if the platform is "web" then -- previously "html5"
    put true into sIsWeb
else
   put false into sIsWeb
end if
```

Any existing code which branches to detect the web platform using this syntax will need to be updated to take this change into account.

New ellipsis token

The character sequence `...` is now reserved as the 'ellipsis' token.

Previously, `...` was allowed in constant, variable and handler names as long as it appeared after the first character. For example, `t...Foo` and `myHandler...` were both valid names.

Such usage will now cause an error so any scripts using such names will need to be updated to accommodate this change.

Web processor identifier changed

The way the web platform's processor is identified has changed to better reflect the use of wasm. rather than pure JavaScript previously.

The return value of `the processor` (and its alternative form `processor()` is now `wasm` instead of `js`.

Example: "' if the processor is "wasm" then -- previously "js" put true into sIsWasm else put false into sIsWasm end if "'

Any existing code which branches to detect the web processor using this syntax will need to be updated to take this change into account.

Correction to system version string on macOS

The engine will now return the correct value for `the systemVersion` on all macOS versions.

Previously, when running on macOS 11.0 and above, the system version would be reported as `10.16.0`.

Now, on Big Sur, the system version will be reported starting from `11.0.0`; and, on Monterey, the system version will be reported starting from `12.0.0`.

Due to this change, any legacy (macOS-specific) scripts which assume that the first number of `the systemVersion` will be `10` to detect running on Mac OS X, or ignore the first number of `the systemVersion` entirely and only compare using the second and third, will have to be updated to accomodate this change.

Other Changes

Android

The IME (soft keyboard on mobile) is now activated and deactivated if required only on redraw. The change allows `lock screen` to delay changes to the IME.

iOS

The IME (soft keyboard on mobile) is now activated and deactivated if required only on redraw. The change allows `lock screen` to delay changes to the IME.

Issues Resolved

Features implemented

23407	**Dictionary and sequence literal syntax has been added allowing much easier expression of array-based values**	**10.0.0-dp-4**
23699	**A new constant `nothing` has been added to expose the special value placed in uninitialized variables**	**10.0.0-dp-4**
23706	**A utility handler has been added to LCB to allow starting Android activities and receiving their results**	**10.0.0-dp-4**
23425	The map widget can now be used in apps deployed to Web	10.0.0-dp-3
	Syntax has been added to LCB allowing encoding to, and decoding from,	10.0.0-

23573	base64	dp-3
23574	Syntax has been added to LCB allowing encoding to, and decoding from, a variety of text encodings	10.0.0-dp-3
2734	A new option `Guidelines` has been added to the `View` menu which turns on dynamic indicators when moving and resizing controls	10.0.0-dp-2
23472	A handler can now be marked as variadic by using `...` as the last declared parameter	10.0.0-dp-2
23479	The Web-specific LCB utility function `HandlerAsJSFunction` can now be used on any handler	10.0.0-dp-2
23480	A new `set` chunk has been added to LCB to allow access to specific custom property sets of script objects	10.0.0-dp-2
23495	Aspect-ratio preserving options have been added to the button's `iconGravity` property	10.0.0-dp-2
23513	Browser widgets now implement the `opaque` property, controlling whether the default background is drawn or not	10.0.0-dp-2
23518	New handlers have been added for easily converting between `JObjectArray` and `List` types in LCB	10.0.0-dp-2
23530	The browser widget can now be used in apps deployed to Web	10.0.0-dp-2
23548	It is now possible to customize the HTML page used to embed apps when deploying to Web	10.0.0-dp-2
23558	A preview of a new chart widget is now available on the tools palette	10.0.0-dp-2
23138	A new `clipSnapshots` property has been added to the barcode scanner on Android	10.0.0-dp-1
23322	Support for GLES3.x `acceleratedRendering` has been added for Android and iOS devices	10.0.0-dp-1
23329	A new bounds option has been added to the LCB `begin layer` command	10.0.0-dp-1
23332	A new `device transform` property has been added to the LCB canvas object	10.0.0-dp-1
23334	A new `is valid` operator has been added to the LCB image object	10.0.0-dp-1
23441	Web deployment now uses WebAssembly	10.0.0-dp-1
22998	Apple architecture support has been added to the macOS IDE, standalone builder, and server engine	pending 9.6.8-rc-1

Regressions fixed

23646	**Using `go` from a popup menu `menuPick` handler will no longer cause an exception when running in web**	**10.0.0-dp-4**
23678	**Deleting a stack will no longer cause an exception when running in web**	**10.0.0-dp-4**

23686	**An execution error no longer occurs when a user clicks on a marker in the Map Widget**	**10.0.0-dp-4**

Bugs fixed

23648	**The Map Widget row of the inclusions pane in the standalone settings now includes Web in the supported platforms list**	**10.0.0-dp-4**
23649	**An issue preventing check for updates working correctly has been resolved**	**10.0.0-dp-4**
23677	**Files created in the web engine's memory filesystem can now be subsequently read from**	**10.0.0-dp-4**
21053	The soft keyboard will now only appear for focused fields when running a web app in a mobile browser	10.0.0-dp-3
23590	The default content type used when doing `POST` or `PUT` requests on web has been corrected to `application/x-www-form-urlencoded`	10.0.0-dp-3
23631	Using `the platform` or `platform()` now returns `web` rather than `html5` when used in Web standalones	10.0.0-dp-3
23636	The processor/architecture of web standalones now reports as `wasm` rather than `js`	10.0.0-dp-3
21666	The browser will no longer automatically launch applications to handle non-standard url schemes on Android	10.0.0-dp-2
23470	The web engine will no longer crash while changing cards when there is a widget with a native layer present	10.0.0-dp-2
23471	The character sequence `...` is now reserved and cannot be used in constant, variable or handler names	10.0.0-dp-2
23476	The character `@` is now reserved and cannot be used as the first character of a constant, variable or handler name	10.0.0-dp-2
23485	URL operations now work on web when the internet library is included	10.0.0-dp-2
23551	The value of `my resources folder` is now correct when LCB extensions are used in apps deployed to Web	10.0.0-dp-2
23565	Controls using native layers on Web now clip correctly when placed in groups	10.0.0-dp-2
16076	The wait command is now supported when using web deployment	10.0.0-dp-1
22840	The IME is now activated and deactivated if required only on redraw	10.0.0-dp-1
22914	The `keyboardActivated` and `keyboardDeactivated` messages now break `wait for messages` on Android.	10.0.0-dp-1
22996	Java methods which trigger LCB handlers no longer cause a JNI error	10.0.0-dp-1
20546	The `revpdfprinter` driver is now included with the server engine allowing `open printing to pdf` to work	pending 9.6.8-rc-1 pending

22650	Repeatedly setting the filename of a mobile player control on iOS now works reliably	9.6.8-rc-1
22887	On macOS, `the systemVersion` will now return the correct value on all operating system versions	pending 9.6.8-rc-1
23515	The default deployment platforms for a new stack are now appropriate for the current license	pending 9.6.8-rc-1
23527	Using `exit to top` in a function called when evaluating a function call's arguments now works correctly	pending 9.6.8-rc-1
23576	AppleScript can now be used to control other applications from both the IDE and standalones on macOS	pending 9.6.8-rc-1
23578	The XPDF external is now available again	pending 9.6.8-rc-1
23579	The minWidth, maxWidth, minHeight and maxHeight properties now work correctly on stacks which have their scaleFactor set	pending 9.6.8-rc-1
23587	Exiting fullscreen media playback will no longer cause a crash on iOS	pending 9.6.8-rc-1
23602	The (legacy) Android app icon standalone setting will now persist from older versions of LiveCode	pending 9.6.8-rc-1
23611	The return value of `do as applescript` is now formatted correctly when running through Rosetta on Apple architecture macs	pending 9.6.8-rc-1
23612	The map widget API key is now reliably included in the standalone manifest	pending 9.6.8-rc-1
23618	Folders added to the `Copy Files` list are now included recursively in web standalones	pending 9.6.8-rc-1
23620	Adding empty files to the `Copy Files` list will no longer cause standalone building to fail when targetting iOS	pending 9.6.8-rc-1
23621	The supported platforms of the `INI Library`, `Mac Status Menu`, `Secure Key Storage`, `Time zone library` and `Drawing library` extensions are now displayed correctly in the standalone builder	pending 9.6.8-rc-1
23626	Encoding text to `utf16be` using the `textEncode` function now works correctly on Windows	pending 9.6.8-rc-1
23638	The user guide PDF no longer has missing images	pending 9.6.8-rc-1
	Using the image variant of `export` when the source is not an image	pending

23655	object will no longer cause a crash.	9.6.8-rc-1
23661	Updating the screen in a tight loop on macOS will no longer cause memory usage to slowly increase	pending 9.6.8-rc-1
23663	The dictionary entries for `mergJSON` are now available again	pending 9.6.8-rc-1
23673	The `put` command can now be used without a target before `else` in a single-line `if` statement	pending 9.6.8-rc-1
23682	Redraw glitches no longer occur when performing a visual effect immediately after another window update on macOS	pending 9.6.8-rc-1
23705	Using the widget variant of `export` when the source is an unknown widget kind will no longer cause a crash	pending 9.6.8-rc-1
12550	The time taken to clone a stack when the message box is open has been reduced	9.6.7
23492	IDE responsiveness will no longer reduce after the message box has been opened and then closed	9.6.7
23531	Script execution errors are now displayed correctly when the script editor is opened in debug mode	9.6.7

Previous release notes

- LiveCode 9.6.7 Release Notes
- LiveCode 9.6.6 Release Notes
- LiveCode 9.6.5 Release Notes
- LiveCode 9.6.4 Release Notes
- LiveCode 9.6.3 Release Notes
- LiveCode 9.6.2 Release Notes
- LiveCode 9.6.1 Release Notes
- LiveCode 9.6.0 Release Notes
- LiveCode 9.5.1 Release Notes
- LiveCode 9.5.0 Release Notes
- LiveCode 9.0.5 Release Notes
- LiveCode 9.0.4 Release Notes
- LiveCode 9.0.3 Release Notes
- LiveCode 9.0.2 Release Notes
- LiveCode 9.0.1 Release Notes
- LiveCode 9.0.0 Release Notes
- LiveCode 8.1.10 Release Notes
- LiveCode 8.1.9 Release Notes
- LiveCode 8.1.8 Release Notes
- LiveCode 8.1.7 Release Notes
- LiveCode 8.1.6 Release Notes

- LiveCode 8.1.5 Release Notes
- LiveCode 8.1.4 Release Notes
- LiveCode 8.1.3 Release Notes
- LiveCode 8.1.2 Release Notes
- LiveCode 8.1.1 Release Notes
- LiveCode 8.1.0 Release Notes
- LiveCode 8.0.2 Release Notes
- LiveCode 8.0.1 Release Notes
- LiveCode 8.0.0 Release Notes
- LiveCode 7.1.4 Release Notes
- LiveCode 7.1.3 Release Notes
- LiveCode 7.1.2 Release Notes
- LiveCode 7.1.1 Release Notes
- LiveCode 7.1.0 Release Notes
- LiveCode 7.0.6 Release Notes
- LiveCode 7.0.4 Release Notes
- LiveCode 7.0.3 Release Notes
- LiveCode 7.0.1 Release Notes
- LiveCode 7.0.0 Release Notes
- LiveCode 6.7.11 Release Notes
- LiveCode 6.7.10 Release Notes
- LiveCode 6.7.9 Release Notes
- LiveCode 6.7.8 Release Notes
- LiveCode 6.7.7 Release Notes
- LiveCode 6.7.6 Release Notes
- LiveCode 6.7.4 Release Notes
- LiveCode 6.7.2 Release Notes
- LiveCode 6.7.1 Release Notes
- LiveCode 6.7.0 Release Notes
- LiveCode 6.6.5 Release Notes
- LiveCode 6.6.4 Release Notes
- LiveCode 6.6.3 Release Notes
- LiveCode 6.6.2 Release Notes
- LiveCode 6.6.1 Release Notes
- LiveCode 6.6.0 Release Notes
- LiveCode 6.5.2 Release Notes
- LiveCode 6.5.1 Release Notes
- LiveCode 6.5.0 Release Notes
- LiveCode 6.1.3 Release Notes
- LiveCode 6.1.2 Release Notes
- LiveCode 6.1.1 Release Notes
- LiveCode 6.1.0 Release Notes
- LiveCode 6.0.2 Release Notes
- LiveCode 6.0.1 Release Notes
- LiveCode 6.0.0 Release Notes