

LiveCode 9.6.0-dp-3 Release Notes

- Overview
- Known issues
- Breaking changes
 - Boolean constants
 - Infinity constant
 - Implicit object
- Platform support
 - Windows
 - Linux
 - Mac
 - iOS
 - Android
 - HTML5
- Setup
 - Installation
 - Uninstallation
 - Reporting installer issues
 - Activating LiveCode Indy or Business edition
 - Command-line installation
 - Command-line uninstallation
 - Command-line activation for LiveCode Indy or Business edition
- LiveCode Community engine changes
 - Update Android text rendering
 - A new output kind `detailed-utf8` has been added to the `files` and `folders` functions
 - Fix iOS mobile browser controls not resizing after setting certain properties.
 - iOS mobile player control updated to use AVKit
 - Mobile browser control iOS update.
 - Print to PDF on Android
 - Specific engine bug fixes (9.6.0-dp-3)
 - Specific engine bug fixes (9.6.0-dp-2)
 - Specific engine bug fixes (9.6.0-dp-1)
- LiveCode Community IDE changes
 - Specific IDE bug fixes (9.6.0-dp-2)
 - Specific IDE bug fixes (9.6.0-dp-1)
- LiveCode Community extension changes
 - SVG Icon widget
 - Browser widget
 - Tree View widget
 - Specific extension bug fixes (9.6.0-dp-3)

- Specific extension bug fixes (9.6.0-dp-2)
- LiveCode Indy engine changes
 - Camera Control Focus Mode
 - cameraControl no longer automatically configures the audio and video input devices when created.
 - Specific engine bug fixes (9.6.0-dp-3)
 - Specific engine bug fixes (9.6.0-dp-2)
- LiveCode Indy extension changes
 - Speech Library
- LiveCode builder changes
 - LiveCode Builder Language
 - LiveCode Builder Host Library
 - Specific builder bug fixes (9.6.0-dp-3)
- Previous 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

Boolean constants

In this release, boolean constants `true` and `false` have been changed so that they resolve to values of boolean type (rather than string). This will affect any uses of the `is strictly` operator on such values, i.e. previously the following were true:

`true is strictly a string` `false is strictly a string`

Now, they are both false, and the following are true:

`true is strictly a boolean` `false is strictly a boolean`

Boolean constants passed as elements of arrays to LCB handlers will not require conversion to boolean values in LCB - in fact any attempt to do so assuming they are strings will cause an error. Any array elements which are intended to be booleans in LCB should be checked for their type before conversion. For example, any of the following could be done by an LCB library user:

```
put true into tArray["enabled"]
put "true" into tArray["enabled"]
put (tVar is not "enabled") into tArray["enabled"]
```

An LCB handler to which `tArray` is passed should do the following:

```
variable tEnabled as Boolean
if tArray["enabled"] is a boolean then
    put tAction["enabled"] into tEnabled
else
    put tAction["enabled"] parsed as boolean into tEnabled
end if
```

Infinity constant

The constant `infinity` has been added to the language in this release. As a result, the unquoted literal `infinity` is now reserved. Any existing uses of it should be quoted, as otherwise it will resolve to the floating point value representing infinity, rather than the string "infinity".

Implicit object

A number of LCB commands use an implicit object to provide context for their execution. Some of these commands also allow specifying an explicit object. These commands are:

- `execute script`
- `send`
- `post`
- `image from file`
- `resolve file` - new in this version

In previous releases `execute script` and `image from file` would use `this card` of the `defaultStack` as the implicit object even if called from a widget. The `send` and `post` commands, however, used `this card` of the `defaultStack` when in a library module handler and the host widget when in a widget module handler. This release changes `execute script` and `image from file` to also use the host widget as the implicit object. This means, for example, that `image from file` will resolve a relative file path relative to the `stackFile` the host widget is on rather than the `stackFile` of the `defaultStack`.

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)
- 10.10.x (Yosemite)
- 10.11.x (El Capitan)
- 10.12.x (Sierra)
- 10.13.x (High Sierra)
- 10.14.x (Mojave)

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 8.2 on MacOS X 10.11
- Xcode 9.2 on MacOS 10.12 (Note: You need to upgrade to 10.12.6)
- 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)

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 8.2* in the *Mobile Support* preferences, to let you test your stack on the *iOS Simulator 10.2*.

We currently support deployment for the following versions of iOS:

- 10.2
- 11.2

- 12.1
- 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 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.x (Nougat)
- 8.x (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	<code><x86 program files folder>/RunRev/LiveCode <version></code>
Linux	<code>/opt/livecode/livecode-<version></code>

The installations when installing for "This User" are:

Platform	Path
Windows	<code><user roaming app data folder>/RunRev/Components/LiveCode <version></code>
Linux	<code>~/.runrev/components/livecode-<version></code>

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 -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.
-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 -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 `.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

Update Android text rendering

This update resolves issues with incorrect spacing between characters when drawing text on Android.

A new output kind `detailed-utf8` has been added to the `files` and `folders` functions

The `{ long | detailed } { files | folders } of <folder>`, `files(<folder>, "detailed")`, and `folders(<folder>, "detailed")` suffer from an anomaly bug where file and folder names are native encoded before being URL encoded to add to the list. The native encoding is can not represent many unicode codepoints and is therefore lossy.

The new `detailed-utf8` output kind encodes file and folder names as utf8 before URL encoding them. This allows the names to be decoded via `textDecode(URLDecode(<name>), "utf8")`.

Fix iOS mobile browser controls not resizing after setting certain properties.

Setting the mobile browser control 'dataDetectorTypes', 'allowsInlineMediaPlayback', and 'mediaPlaybackRequiresUserAction' properties will no longer prevent the underlying view from resizing to fit the configured rect of the browser control.

iOS mobile player control updated to use AVKit

The existing implementation of the mobile player control for iOS has been replaced with one based on the AVKit framework.

While every attempt has been made to ensure full compatibility with the previous implementation, differences between the frameworks used means that some features may not function as before or may no longer be available. These are listed below:

- `mobileControlDo` actions: **"begin seeking forward" - unsupported** "begin seeking backward" - unsupported ** "end seeking" - unsupported
- `mobileControlSet` properties: **"useApplicationAudioSession" - unsupported** "allowsAirPlay" - now covers all external playback
- `mobileControlGet` properties: ** "playbackState" - values "seeking forward" and "seeking backward" unsupported
- Messages: **"playerEnterFullscreen" - enter fullscreen via user action not detected** "playerLeaveFullscreen" - leave fullscreen via user action not detected ** "playerMovieChanged" - unsupported

Mobile browser control iOS update.

The mobile browser control has been updated to use the WebKit framework on iOS. This provides better performance on iOS devices.

Print to PDF on Android

The `open printing to pdf ...` command can now be used to direct printing to a PDF output file on Android devices.

Specific engine bug fixes (9.6.0-dp-3)

- 7948** Added note that `revDeleteFolder` can not delete the `defaultFolder`
- 16240** Explained the non-recursive behaviour of the `replace` command in its entry.
- 17361** Ensure `mobileControlDo "myinput", "focus"` works on Android
- 18876** Fix text duplication when deleting text previously entered into a field on Android
- 19997** Ensure "the `printPaperRectangle`" returns the correct value
- 22295** Added support for splash screens and icon for iPad Pro 11
- 22508** Fix clipboard contents being cleared when setting `dragdata` on Windows
- 22511** Fix multi-codepoint character parsing in `JSONImport()`
- 22530** Update Android text rendering
- 22533** Add support for the `layerClipRect` property to widgets
- 22536** The `replace` command will no longer fail incorrectly in some cases where there are no possible replacements to be made
- 22552** Fix crash touching screen during `preOpenStack` on Android
- 22556** Drag and drop on macOS from LiveCode to another application no longer fails after three drags
- 22577** Ensure the device token in push notifications is returned correctly

- 22591** Ensure drag-n-drop of a filename with spaces onto Mail app adds the file as an attachment
- 22593** Fix support for font smoothing in macOS 10.14+
- 22602** Ensure setting the keyboardType has an effect on iOS even if the returnKeyType is not set
- 22612** Allow detection of "Apple Distribution" certificates
- 22614** Ensure focused control is correct when opacity of control under mouse pointer changes
- 22615** Fix error when building an Android standalone if the Barcode Scanner widget is included
- 22616** Fixed browser widget within a group on Android appearing in an incorrect position after moving the group

Specific engine bug fixes (9.6.0-dp-2)

- 22190** Ensure setting the spaceAbove does not change the existing value of spaceBelow property
- 22213** A new output kind `detailed-utf8` has been added to the `files` and `folders` functions
- 22279** Fix page not displayed after setting `htmltext` of browser widget
- 22378** Ensure text sort of mixed ASCII/Unicode lines of a variable works correctly
- 22388** Add support for merging activity attributes in Android Manifest Merging mechanism
- 22412** Fix `tabAlign` property not saving when `tabStops` is not set
- 22457** Prevent crash when calling `binaryDecode` with wrong data input
- 22483** Fix iOS mobile browser controls not resizing after setting certain properties.
- 22497** Fix listing the executing object's stack rather than the object in the `executionContexts` for password protected stacks

Specific engine bug fixes (9.6.0-dp-1)

- 22278** Corrected missing text issue in the documentation for the log command
- 22346** Fix crash when an error occurs in a SQLite query
- 22382** Fix incorrect stack size on Android for fullscreen mode `showAll` when the device has a soft button bar
- 22414** Fix crash on Android coalescing window reshape events
- 22463** Fix compilation of ellipse elements in drawing svg compiler

LiveCode Community IDE changes

Specific IDE bug fixes (9.6.0-dp-2)

- 22092 Sync auto complete field when msg box is scrolled horizontally
- 22137 Fix issue causing conditional breakpoints in repeat loops to only be evaluated once
- 22145 Add missing variable declaration to editorcommon
- 22477 Fix broken links in Resource Center
- 22486 Fix missing return in commoneditor scriptFormat

Specific IDE bug fixes (9.6.0-dp-1)

- 16933 Adapt icon size to text size in Project Browser
- 21576 Split non-synonymous DataGrid commands deleteIndex and deleteIndexes as well as deleteLine and deleteLines in the documentation.
- 22440 Fix card indentation in PB if stack name contains "of"
- 22443 Align widget "icon" with connectors in Project Browser

LiveCode Community extension changes

SVG Icon widget

Implementation

The svgpath widget will now only perform costly scaledWidth and scaledHeight recalculations when the widget width or height changes.

Browser widget

Browser widget iOS update.

The browser widget has been updated to use the WebKit framework on iOS. This provides better performance on iOS devices.

In addition, the update enables the browser widget 'userAgent' property on iOS.

Tree View widget

Properties

Enhance the Tree View Widget to support use on the mobile platform.

- Adjust row height when font name or size changes
- Trimming improvements
 - When font name or size changes, re-evaluate ellipsis width to properly trim contents of keys/values

- In `readOnly` mode, only save space for a single icon on the right
- Prevent last character from being turned into an ellipsis
- Width was being reported with extra space for an ellipsis when not needed
- Remove padding from ellipsis width, only add padding when needed
- Save `alternate row backgrounds` property
- Save `show border` property

New properties:

- `charsToTrimFromKey` - allow a sorting value to be added to the front of the key that is trimmed for display
- `hilitedElementIsFolded` - adjust the fold state of the selected element
- `formattedHeight` - content height for scroller support
- `scroll` and `vScroll` - scroll position
- `textHeight` - custom row height
- `vScrollBar` - control visibility of scroll bar
- `showHover` - allow hover to be disabled, useful on mobile
- `iconHeight` - allow configuration of icon size
- `showValues` - allow the values to be hidden

Signals

- `formattedHeightChanged` - message to report content height change to support scrollers

Specific extension bug fixes (9.6.0-dp-3)

22509 Ensure null `mEllipsisLength` does not cause widget to crash

Specific extension bug fixes (9.6.0-dp-2)

21719 Allow setting fold state of selected element

22241 Ensure getting the `scrollingEnabled` of the native android field does not throw an error

22386 Save `alternate row backgrounds` property

22387 Save `showBorder` property

22498 Fixed a bug in the compilation of SVG `'rotate()'` transforms

LiveCode Indy engine changes

Camera Control Focus Mode

The `focusModes` and `focusMode` properties can now be used with `cameraControlGet` and `cameraControlSet` on Android devices to list and choose from the modes available on the configured camera input device.

The existing modes `fixed`, `auto`, and `continuous` together with new modes `autoSmooth`, `macro`, `macroSmooth`, `infinity`, `continuousSmooth`, and `extended` are now supported on Android, and are described in the dictionary entries for `cameraControlGet` and `cameraControlSet`.

The `autoSmooth`, `macro`, `macroSmooth`, and `continuousSmooth` focus modes are also now available on iOS.

cameraControl no longer automatically configures the audio and video input devices when created.

This allows the cameraControl to be configured for audio-only or video-only recording without unnecessary permission request dialogs being presented when the cameraControl is created.

The "videoDevices", "audioDevices", "videoDevice", and "audioDevice" cameraControl properties have been made available on Android, allowing video and audio input devices to be configured independently.

The "videoDevice" property accepts the values "default", "front", and "back" to specify the corresponding video input device, and the "audioDevice" property accepts the value "default" to specify the default audio input device.

You can restore the previous behaviour by adding two more lines to your scripts to set both video and audio devices to their defaults:

```
cameraControlCreate "myCamera"
cameraControlSet "myCamera", "videoDevice", "default"
cameraControlSet "myCamera", "audioDevice", "default"
```

Specific engine bug fixes (9.6.0-dp-3)

- 22409** Fix unusable files created by cameraControl when recording more than 10 seconds of audio on Mac
- 22540** Improve quality of WMV video files recorded by cameraControl on Windows

Specific engine bug fixes (9.6.0-dp-2)

- 18655** Fix handling of device orientation in iOS and Android camera controls

LiveCode Indy extension changes

Speech Library

New library to synthesize speech

A new macOS, iOS and Android library has been added that provides functionality for synthesizing speech has been implemented.

The library allows for choosing from among the system voices, altering the speech rate and pitch of the voice and speaking an utterance.

See the dictionary for full details.

LiveCode builder changes

LiveCode Builder Language

Objective-C block support

The handler `CreateObjcBlockPointerFromHandler` has been added the `objc` module which facilitates the calling of Objective-C methods that require blocks. Objective-C blocks are objects defining fragments of code which can be called at a later date, typically used by methods that require progress callbacks or completion handlers.

`CreateObjcBlockPointerFromHandler` wraps a LCB handler in an `ObjcBlockPointer` and returns true on success and false otherwise. In order to call the `CreateObjcBlockPointerFromHandler`, pass the handler to be wrapped as the first argument and the variable into which the `ObjcBlockPointer` should be placed as the second.

```

private variable sRequestPermissionsCompletionHandler as optional
ObjcBlockPointer
private variable sTarget as ScriptObject

public handler AudioLibraryInitialize() returns Boolean
  if not
    CreateObjcBlockPointerFromHandler(RequestPermissionsCompletionHandler, \
sRequestPermissionsCompletionHandler) then
      put nothing into sRequestPermissionsCompletionHandler
      return false
    end if
    put the caller into sTarget
    return true
  end handler

public handler RequestPermissionsCompletionHandler(in pBlock as
ObjcBlockPointer, in pGranted as CBool)
  post "AudioLibraryRequestPermissionsCallback" to sTarget with [pGranted]
end handler

```

In the example above, the handler `RequestPermissionsCompletionHandler` is wrapped by the `ObjcBlockPointer` `sRequestPermissionsCompletionHandler`.

The first parameter of the wrapped handler should be an `ObjcBlockPointer`. The remaining parameters should match those of the Objective-C block the `ObjcBlockPointer` will be used with.

Once created, an `ObjcBlockPointer` can be used to call Objective-C methods that require blocks, passing the created `ObjcBlockPointer` where an Objective-C block would be expected.

```

private foreign handler ObjC_AVCaptureDeviceRequestAccessForMediaType(in
pMediaType as ObjcId, in pCompletionHandler as ObjcBlockPointer) \
  returns nothing \
  binds to
"objc:AVCaptureDevice.+requestAccessForMediaType:completionHandler:"

public handler AudioLibraryRequestPermissions()
  unsafe

  ObjC_AVCaptureDeviceRequestAccessForMediaType(StringToNSString("soun"),
sRequestPermissionsCompletionHandler)
  end unsafe
end handler

```

In the example above, the handler `AudioLibraryRequestPermissions` uses the previously

created block pointer, `sRequestPermissionsCompletionHandler`, when calling the `requestAccessForMediaType:completionHandler:` method of the `AVCaptureDevice` type. The method expects the second argument, `completionHandler:`, to be a block that takes a single `BOOL` parameter, which in this case matched the signature of the handler `RequestPermissionsCompletionHandler`.

The wrapped handler will be called whenever the block is invoked, with the first parameter being the `ObjcBlockPointer` used to wrap the handler.

In the above example, the `RequestPermissionsCompletionHandler` handler will be called when the `completionHandler:` block is invoked, which in this case will be when the request permissions process has completed.

The lifetime of a created `ObjcBlockPointer` is not automatically managed. When such a value has no more references to it and it is no longer going to be used, `DeleteObjcBlockPointer` should be used to free the resources used by it.

```
public handler AudioLibraryFinalize()
  if sRequestPermissionsCompletionHandler is not nothing then
    DeleteObjcBlockPointer(sRequestPermissionsCompletionHandler)
    put nothing into sRequestPermissionsCompletionHandler
  end if
end handler
```

LiveCode Builder Host Library

Widget library

The `popup widget` command now correctly positions the widget.

Specific builder bug fixes (9.6.0-dp-3)

19973 A `null` listener parameter no longer causes a crash.

22603 The `popup widget` command incorrectly positions the popped up widget

Previous 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 8.1.9 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 8.1.10 Release Notes](#)
- [LiveCode 8.1.1 Release Notes](#)
- [LiveCode 8.1.0 Release Notes](#)
- [LiveCode 8.0.2 Release Notes](#)
- [LiveCode 8.0.1 Release Notes](#)
- [LiveCode 8.0.0 Release Notes](#)
- [LiveCode 7.1.4 Release Notes](#)
- [LiveCode 7.1.3 Release Notes](#)
- [LiveCode 7.1.2 Release Notes](#)
- [LiveCode 7.1.1 Release Notes](#)
- [LiveCode 7.1.0 Release Notes](#)
- [LiveCode 7.0.6 Release Notes](#)
- [LiveCode 7.0.4 Release Notes](#)
- [LiveCode 7.0.3 Release Notes](#)
- [LiveCode 7.0.1 Release Notes](#)
- [LiveCode 7.0.0 Release Notes](#)
- [LiveCode 6.7.9 Release Notes](#)
- [LiveCode 6.7.8 Release Notes](#)
- [LiveCode 6.7.7 Release Notes](#)
- [LiveCode 6.7.6 Release Notes](#)
- [LiveCode 6.7.4 Release Notes](#)
- [LiveCode 6.7.2 Release Notes](#)
- [LiveCode 6.7.11 Release Notes](#)
- [LiveCode 6.7.10 Release Notes](#)
- [LiveCode 6.7.1 Release Notes](#)
- [LiveCode 6.7.0 Release Notes](#)
- [LiveCode 6.6.5 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](#)