LiveCode 8.0.0-dp-3 Release Notes

Table of contents

Overview
- I don't want to build extensions. What's in it for me?
- LiveCode Script vs LiveCode Builder
- Warning
- IDE

Known issues

Platform support
- Windows
- Linux
- Mac

Setup
- Installation
- Uninstallation

Reporting installer issues

Activation

Multi-user and network install support (4.5.3)

Command-line installation

Command-line activation

Engine changes
- Image metadata
- LiveCode Builder
  - LiveCode Builder Language
  - Extensions
  - Getting Started
- Packaged extensions naming consistency
- Feature: Popup Widgets
  - New Syntax:
- Various bugs with navigation bar widget
- Specific bug fixes (8.0.0-dp-3)
- Specific bug fixes (8.0.0-dp-1)

IDE changes
- Menu bar
- Property Inspector
- Widget metadata and the IDE
- Standalone Settings
- Property Inspector
  - Property Attributes
    - default
    - editor
    - group
    - label
    - options
    - section
    - user_visible
    - read_only
  - Widget Properties
  - Script Object Properties
  - Editors

- IDE stackfiles named with version.
- Specific bug fixes (8.0.0-dp-3)
- Specific bug fixes (8.0.0-dp-1)

LiveCode Builder changes

- LiveCode Builder Tools
  - Compiler generates an error if integer literal too big
  - lc-compile
    - Command-line interface
    - Warnings
  - Bugs fixed
- LiveCode Builder Language
  - Case-Sensitivity
  - Foreign handler definitions require explicit typing.
  - Replace concept of 'undefined' with 'nothing'
  - IntSize Type
  - Change to handler return type syntax.
  - Syntax
  - Identifiers
  - Bugs fixed
- LiveCode Builder Host Library
  - Ability to access a widget's effective font
  - Determining if a widget is enabled
  - Widget Printing
  - Native Code Access
  - Ability to display a popup menu
  - Detecting successive clicks
  - Composed widgets
    - Example
    - Syntax
    - Events
    - Messages
  - Mathematical functions

Dictionary additions

Dictionary changes

Previous Release Notes

Overview

LiveCode 8.0 is the most exciting release in the history of the technology. It provides a simple way to extend the functionality or control set of LiveCode.

Our focus in LiveCode 8.0 is extensibility. You can now build and share widgets (custom controls) and libraries that are treated by LiveCode as engine level elements.

LiveCode 8.0 can be thought of as a version 7.0 with a new module allowing extensions to be plugged into the engine. As a result, 8.0 should be as functional and stable as LiveCode 7.0.

I don't want to build extensions. What's in it for me?

Many love LiveCode because of the productivity benefits and donâ€™t have time to build extensions. If that is the case just kick back and start using LiveCode 8 and keep an eye on the extensions portal. You can start using new controls and libraries as they are built by other community members.

LiveCode Script vs LiveCode Builder

To make it possible to create extensions and plug them into the LiveCode engine we've created a new flavour of our language called . LiveCode Builder looks a lot like LiveCode Script so should feel familiar for any seasoned LiveCode developer. There is lots of new syntax which exposes parts of the LiveCode engine that were only previously available to those who were skilled c/c++ developers.

LiveCode Builder is a new language and is therefore highly experimental and should be considered an early prototype. It will take some getting used to but we know you'll love it once you see how powerful it is. The best way to get started is to read the "Extending LiveCode" guide which can be found in the dictionary under the "Guide" tab.

Warning

It is important to stress that **no aspect of this release should be considered final**. **Every piece of syntax in LiveCode Builder is subject to change**

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 new property inspector lacks some properties present in the old property inspector
- The supplied widgets are examples and lack features and general robustness
- The extension builder plugin "Test" feature fails if the widget being tested is already installed - uninstalling the widget and restarting the IDE should help
- All installed widgets are built into any standalones produced

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

The engine supports the following Windows OSes:

- Windows XP SP2 and above
- Windows Server 2003

- Windows Vista SP1 and above (both 32-bit and 64-bit)
- Windows 7 (both 32-bit and 64-bit)
- Windows Server 2008
- Windows 8.x (Desktop)

Note: *On 64-bit platforms the engine still runs as a 32-bit application through the WoW layer.*

Linux

The linux engine requires the following:

- Supported architectures:

32-bit or 64-bit Intel/AMD or compatible processor
32-bit ARMv6 with hardware floating-point (e.g. RaspberryPi)

- Common requirements for GUI functionality:

GTK/GDK/Glib 2.24 or later
Pango with Xft support
(optional) *esd - required for audio output*
(optional) mplayer - required for media player functionality
(optional) *lcms - required for color profile support in images*
(optional) gksu - required for privilege elevation support

- Requirements for 32-bit Intel/AMD:

glibc 2.3.6 or later

- *Requirements for 64-bit Intel/AMD:*

glibc 2.15 or later

- Requirements for ARMv6:

glibc 2.7 or later
Note: The GUI requirements are also required by Firefox and Chrome, so if your Linux distritution runes one of those, it will run the engine.
Note: If the optional requirements are not present then the engine will still run but the specified features will be disabled.
Note: It may be possible to compile and run LiveCode Community on other architectures but this is not officially supported.

Mac

The Mac engine supports:

- *10.6.x (Snow Leopard) on Intel*
- *10.7.x (Lion) on Intel*
- *10.8.x (Mountain Lion) on Intel*
- *10.9.x (Mavericks) on Intel*

Note: The engine runs as a 32-bit application regardless of the capabilities of the underlying processor.

Setup

Installation

Each distinct version has its own complete folder – multiple versions will no longer install side-by-side: on Windows (and Linux), each distinct version will gain its own start menu (application menu) entry; on Mac, each distinct version will have its own app bundle.
The default location for the install on the different platforms when installing for 'all users' are:

- Windows: <x86 program files folder>/RunRev/ LiveCode 8.0.0-dp-3
- Linux: /opt/runrev/livecode-8.0.0-dp-3
- Mac: /Applications/ LiveCode 8.0.0-dp-3.app

The default location for the install on the different platforms when installing for 'this user' are:

- Windows: <user roaming app data folder>/RunRev/Components/LiveCode 8.0.0-dp-3
- Linux: ~/.runrev/components/livecode-8.0.0-dp-3
- Mac: ~/Applications/ LiveCode 8.0.0-dp-3.app

Note: *If your linux distribution does not have the necessary support for authentication (gksu) then the installer will run without admin privileges so you will have to manually run it from an admin account to install into a privileged location.*

Uninstallation

On Windows, the installer hooks into the standard Windows uninstall mechanism. This is accessible from the appropriate pane in the control panel.
On Mac, simply drag the app bundle to the Trash.
On Linux, the situation is currently less than ideal:

- open a terminal
- *cd* to the folder containing your rev install. e.g.

cd /opt/runrev/livecode-8.0.0-dp-3

- execute the *.setup.x86* file. i.e.

./.setup.x86

- follow the on-screen instructions.

Reporting installer issues

If you find that the installer fails to work for you then please file a bug report in the RQCC or email support@runrev.com so we can look into the problem.
In the case of failed install it is vitally important that you include the following information:

- Your platform and operating system version
- The location of your home/user folder
- The type of user account you are using (guest, restricted, admin etc.)
- The installer log file located as follows:
- **Windows 2000/XP:** <documents and settings folder>/<user>/Local Settings/
- **Windows Vista/7:** <users folder>/<user>/AppData/Local/RunRev/Logs
- **Linux:** <home>/.runrev/logs
- **Mac:** <home>/Library/Application Support/Logs/RunRev

Activation

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.

Multi-user and network install support (4.5.3)

In order to better support institutions needing to both deploy the IDE to many machines and to license them for all users on a given machine, a number of facilities have been added which are accessible by using the command-line.
Note: *These features are intended for use by IT administrators for the purposes of deploying LiveCode in multi-user situations. They are not supported for general use.*

Command-line installation

It is possible to invoke the installer from the command-line on both Mac and Windows. When invoked in this fashion, no GUI will be displayed, configuration being supplied by arguments passed to the installer.
On both platforms, the command is of the following form:
<exe> install noui *options*
Here *options* is optional and consists of one or more of the following:

-allusers	Install the IDE for all users. If not specified, the install will be done 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 location to install into. If not specified, the location defaults to those described in the *Layout* section above.
-log *logfile*	A file to place a log of all actions in. If not specified, no log is generated.

Note that the command-line variant of the installer does not do any authentication. Thus, if you wish to install to an admin-only location you will need to be running as administrator before executing the command.
As the installer is actually a GUI application, it needs to be run slightly differently from other command-line programs.
In what follows <installerexe> should be replaced with the path of the installer executable or app (inside the DMG) that has been downloaded.
On Windows, you need to do:
start /wait <installerexe> install noui *options*
On Mac, you need to do:
“<installerexe>/Contents/MacOS/installer” install noui *options*
On both platforms, the result of the installation will be written to the console.

Command-line activation

In a similar vein to installation, it is possible to activate an installation of LiveCode for all-users of that machine by using the command-line. When invoked in this fashion, no GUI will be displayed, activation being controlled by any arguments passed.

On both platforms, the command is of the form:
<exe> activate -file *license* -passphrase *phrase*
This command will load the manual activation file from *license*, decrypt it using the given *passphrase* and then install a license file for all users of the computer. Manual activation files can be downloaded from the 'My Products' section of the RunRev customer accounts area.
This action can be undone using the following command:
<exe> deactivate
Again, as the LiveCode executable is actually a GUI application it needs to be run slightly differently from other command-line programs.
In what follows <livecodeexe> should be replaced with the path to the installed LiveCode executable or app that has been previously installed.
On Windows, you need to do:
start /wait <livecodeexe> activate -file *license* -passphrase *phrase*
start /wait <livecodeexe> deactivate
On Mac, you need to do:
“<livecodeexe>/Contents/MacOS/LiveCode” activate -file *license* -passphrase *phrase*
“<livecodeexe>/Contents/MacOS/LiveCode” deactivate
On both platforms, the result of the activation will be written to the console.

Engine changes

Image metadata (8.0.0-dp-3)

A new read only image property has been added to access the metadata in the image file. The returned array is in the same format as that used for the export command. If no metadata is found then the property returns empty rather than an array with empty elements. Currently the only metadata key that is implemented is density which can be used to determine pixel density in pixels per inch. Metadata is currently only parsed from JPEG and PNG file formats.

For example:

```
put the metadata of image 1 into metadataArray

set the width of image 1 to the width of image 1 div (medatadaArray["density"] / 72)
set the height of image 1 to the height of image 1 div (medatadaArray["density"] / 72)
```

LiveCode Builder (8.0.0-dp-3)

LiveCode Builder Language

LiveCode Builder is a variant of the current LiveCode scripting language (LiveCode Script) which has been designed for 'systems' building. It is statically compiled with optional static typing and direct foreign code interconnect (allowing easy access to APIs written in other languages). The compiled bytecode can then be packaged together with any required resources (icons, documentation, images, etc) into a .lce extension package.

Unlike most languages, LiveCode Builder (LCB) has been designed around the idea of extensible syntax. Indeed, the core language is very small - comprising declarations and control structures - with the majority of the language syntax and functionality being defined in modules.

Note: It is an eventual aim that control structures will also be extensible, however this is not the case in the current incarnation).

The syntax will be familiar to anyone who has coded with LiveCode Script, however LiveCode Builder is a great deal more strict - the reason being it is intended that it will eventually be compilable to machine code with the performance and efficiency you'd expect from any 'traditional' programming language. Indeed, over time we hope to move the majority of implementation of the whole LiveCode system over to being written in LiveCode Builder.

Note: One of the principal differences is that type conversion is strict - there is no automatic conversion between different types such as between number and string. Such conversion must be explicitly specified using syntax (currently this is done using syntax like *... parsed as number* and *... formatted as string*.

Extensions

There are two types of extensions which can be written in LCB: widgets and libraries. All installed extensions appear in the new Extension Manager stack, which can be opened from the Tools menu.

An LCB library is a new way of adding functions to the LiveCode message path. Public handlers in loaded LCB libraries are available to call from LiveCode Script.

A widget is a new type of custom control which, once compiled and packaged, can be loaded into the IDE. Using the widget is no different from any of the classic LiveCode controls you've been used to. Simply drag it onto a stack and start interacting with it as you would any another control.

You can reference the widget in script as a control:

```
set the name of the last control to "clock"
```

Or more specifically as a widget:

set the tooltip of widget 1 to "This is my nice new clock widget"

Getting Started

To get started with LiveCode Builder, click on the "Dictionary" icon in the IDE toolbar, select the "Guide" tab and then "Extending LiveCode" from the drop-down menu. This will show you the user-guide on getting started with writing widgets and libraries in LCB. Alternatively, you can start by looking at some of the extensions shipped with LiveCode 8 - the source and other resources for these are located in the "extensions" sub-folder of your LiveCode installation directory (source files are named

Packaged extensions naming consistency (8.0.0-dp-3)

Earlier versions of the widgets and libraries which are bundled with the IDE were named inconsistently.

Now all LiveCode extensions are named either com.livecode.widget.<widget name> or com.livecode.library.<library name>.

Note: This change will break some stacks that have widgets saved on them, or scripts which refer to a widget by its kind.

Feature: Popup Widgets (8.0.0-dp-3)

Added the ability to use widgets within popup dialog windows.

New Syntax:

- popup widget <Kind> at <Position> [with properties <Properties>]
- Launch the named widget as a popup. The popup can return a value in the result.
- currently popped up
- test if this widget is part of a popup
- close popup [returning <Result>]
- set the result of the calling popup statement to <Result>

Various bugs with navigation bar widget (8.0.0-dp-3)

- Selecting "names" as the itemStyle does not work
- Changing the navIcons via script / property inspector does not work
- The navSelectedIcons property is missing.
- editMode should default to false
- add 'navigate' message to widget docs

Specific bug fixes (8.0.0-dp-3)

(bug fixes specific to the current build are highlighted in bold, reverted bug fixes are stricken through)

15743 iOS standalone engine do not build anymore in Debug mode
15723 Incorrect wording in Business edition activation screen
15719 An error in a preOpenStack script aborted openStack
15718 'SSL library not found' error thrown on iOS when using SSL & Encryption library
15685 Fix the path to the OSX standalone engine on Linux
15681 Occasional issue parsing SVG data in LCB
15630 get property tVar of my script object not working in develop branch
15620 Check if m_rep is nil in MCImage::GetMetadataProperty
15618 Codeunit and delimited chunk offsets probably broken in 8.0
Property inspector does not update when graphics are being created using

15605 tools
15509 Condition " in " does not throw parse error for 'case' or 'repeat until/while'
15405 LCS: Can't create a widget in a group
15378 backColor doesn't work properly for graphic on closed stack
15358 LC 8 has a very noisy startup
15286 Palette Actions: Nav items need tooltips
15224 Various bugs with navigation bar widget
15214 IDE-Widgets: Icon picker does odd things when resized
15156 Putting value into item of empty variable hangs LiveCode
15056 Read from file for (x | x chars | x bytes) returns empty
14996 LCB-Canvas: polyline path
14961 Gradient - Quality set to "good" makes LC crash
14806 LCB-Canvas: curve through examples are incorrect in docs

Specific bug fixes (8.0.0-dp-1)

14851 Popup won't stop displaying when displayed in mouseDown of button widget
14602 URLEncode crashes LiveCode
14599 LCB: Text sort is inconsistent with string comparison
14538 bool formatted as string does not work

IDE changes

Menu bar (8.0.0-dp-3)

The menubar has been made a script-only stack to facilitate bugfixes and community contributions. Users should not notice much difference in terms of its appearance. Some of the menu items have been changed, however:

The 'New Mainstack' item now has a submenu with a range of size choices, as well as the option to create a script-only stack. Selecting script-only stack will prompt a choice of name, and subsequently open the stack in the script editor.

We have centralised the building and handling of contextual menus in the menubar script, thereby making per-object contextual menus display and behave consistently throughout the IDE.

The Object > New Control submenu is now generated based on the property information present for each object type, and the newly added Object > New Widget submenu is generated based on the currently
loaded widget extensions.

Property Inspector

A number of changes have been made to property editors in the property inspector:
- The color editors now use a color swatch widget to display the chosen color
- Numeric editors have a slider if the property has an associated min/max, and an increment/decrement twiddle if it has a step value.
- The navbar widget now uses a version of itself as an editor for its properties (com.livecode.pi.navbar)
- A graphic effects property editor has been added (com.livecode.pi.graphiceffect)
- A gradient property editor has been added (com.livecode.pi.gradientramp)
- A script property editor has been added, which contains a button to edit the selected script (com.livecode.pi.script)
- A time zone property editor has been added, which contains a drop-down list of time zones (com.livecode.pi.timezone)

Widget metadata and the IDE

Widget metadata now controls a number of additional features with respect to how the widget interacts with the IDE.
Firstly, the preferredsize attribute controls the initial size of the widget when dragged out from the tools palette.
For example, the navbar widget now has

metadata preferredSize is "320,49"

so that when dragged out, it is created at the correct size for an original iPhone screen.

Secondly, the uservisible attribute controls whether the widget appears at all in the tools palette of the IDE.
A number of widgets have been declared user invisible for this release, either because they are not meant to be
draggable objects at all (eg the icon picker widget, which is designed to be popped up) or are not quite refined
to the point where they are suitable for user stacks, but are included because they are being used in the IDE
(for example the tree view widget).

Finally if present, the svgicon attribute will be used to display an icon for the widget in the tools palette, taking precedence over the included icon resources. All of the widgets included by default in the tools palette now use
svg icon paths.

Standalone Settings

A field has been added to the Copy Files tab of the standalone settings which is populated with the list of currently
installed extensions. All selected extensions from this list are included in standalones and loaded when the standalone
is launched. 'Use' dependencies are automatically calculated and included along with the top-level widget.

Property Inspector (8.0.0-dp-3)

The property inspector has been rewritten to allow properties of widgets to be inspected and edited. It has been implemented with flexibility and extensibility in mind, since it must be able to control the values of widget properties in any way required by the widget developer. Each property now has a number of attributes which affect how it appears in the inspector.

Property Attributes

The following is the list of property attributes:

default

The default value of the property. If there is no default value (for example the 'loc' property does not have one), the string "no_default" can be used. The property inspector pops up a contextual menu when editors are right-clicked allowing the user to set the property back to a default value.

editor

The editor that will be used to display the value of the property and allow it to be edited. See the dedicated section below for details on property inspector editors.

group

Properties are grouped by themselves in the inspector by default. If a particular group name is specified for a set of properties, their editors are placed next to each other in the inspector.

label

The label to use for this property.

options

For properties whose value is a choice from a set of options, that set should be specified as a comma delimited list for the options attribute. Default editors are provided for 'enum' type properties (choice of exactly one from a set) and 'set' type properties (choice of zero or more from a set).

Lists of options can be generated using LiveCode Script for the inspector at run-time, by using the 'execute' syntax - for example the options for the textFont property are generated using

execute: get the fontNames; sort it

Whatever remains in the 'it' variable after executing the specified script is used as the list of options.

section

The section attribute controls which tab of the property inspector contains the property in question. Currently this is required to be one of the following

- Basic
- Data Grid
- Custom
- Table
- Colors
- Effects
- Icons
- Position
- Text

But in the future it may be possible to specify custom sections.

user_visible

Properties are visible in the property inspector by default. Set the user_visible attribute to false to hide a given property from the user.

read_only

Read only properties will be displayed in the property inspector but the corresponding editor will have its "editorEnabled" property set to false. See the Editors section below for more details on enabled/disabled editors.

Widget Properties

Widget metadata is used to control the display of widget properties in the inspector. Items of metadata which determine property attributes are of the form:

metadata <property>.<attribute> is "<value>"

These are stored as property data for the widget at load time. The <attribute> can be any of those specified in the Property Attributes section above. If the attributes are not specified, their values are as follows:

- default - "no_default"
- editor - "com.livecode.pi.number" for Integer/Real properties, "com.livecode.pi.<type>" for properties of type <type>.
- group - the name of the property
- label - the name of the property
- options - empty
- section - "Basic"
- user_visible - true
- read_only - true if there is no specified 'set' handler or variable for the property, false otherwise.

Script Object Properties

Script-level properties of objects (including widgets) are specified in files in the Toolset/resources/supporting_files/property_definitions folder. The propertyInfo.txt file specifies the default values for all the property attributes. Each object type then has a specification of which properties should be displayed in the inspector when it is the selected object, and any options/default/group values which override the defaults.

Editors

Currently an editor must be a stack consisting of a group named "template" and a button named "behavior". The property inspector looks up the specified editor for a given property, clones the template group, and sets its behavior to the long id of the button.

The behavior script must at a minimum implement the following three handlers:

on editorInitialize
on editorUpdate
on editorResize

There are a number of properties available to any editor:

- editorMinWidth
- editorMaxWidth
- editorEnabled
- editorEffective
- editorValue

These should be set or got appropriately. For example, if an editor consists of a text field, the editorUpdate handler should update the value of the field with 'the editorValue of me'. Similarly, if the content of the field changes, the field should call a function in the behavior which sets 'the editorValue of me' to the content of the field.

The editorEnabled and editorEffective properties are set by the generic behavior depending on the property info and the values of the properties. The editorEffective is true if the value of the property in question is empty but there is an effective value in play. The editor should alter the display of its value accordingly.

Editors can specify their min and max width if required.

The following editors are built-in, and available to use for widget properties with common types:

- com.livecode.pi.array - a Tree View widget
- com.livecode.pi.boolean - a check box
- com.livecode.pi.color - a color swatch and dialog
- com.livecode.pi.colorwithalpha - a color swatch and dialog, and alpha value slider
- com.livecode.pi.enum - an option menu

- com.livecode.pi.file - a file selector
- com.livecode.pi.number - a single-line field with increment/decrement twiddle
- com.livecode.pi.pattern - a pattern selector
- com.livecode.pi.set - a field with multi-select list behavior
- com.livecode.pi.string - a single-line field
- com.livecode.pi.text - a multi-line field

There are also some bespoke editors for particular object properties:

- com.livecode.pi.customprops
- com.livecode.pi.datagrid
- com.livecode.pi.textalign
- com.livecode.pi.textstyle

It is our intention that ultimately a widget alone will be able to function as a property editor, however currently this feature is not available.

IDE stackfiles named with version. (8.0.0-dp-1)

When a binary stackfile is rewritten in the IDE for a new version, it should have a (major) version in the filename to prevent unwanted IDE merging between versions. This can also be used to ensure incompatible stacks are not loaded if present - the IDE will only load stacks with a version less than or equal to its version.

For example, from 8.0 onwards, revTools has filename /Toolset/revTools.8.rev.

Specific bug fixes (8.0.0-dp-3)

(bug fixes specific to the current build are highlighted in bold, reverted bug fixes are stricken through)

15601	**traversalOn property missing from widget property inspector**
15600	**Array and enum return values not displayed correctly in dictionary**
15542	**Typo in stack property inspector in LC 8.0**
15464	**Browse tool selected when launching LC8**
15430	**PI doesn't show foregroundColor for legacy graphic control**
15427	**Cut**
15323	**Style property should be an enum**
15293	**Behavior property inspector control should have way to edit behavior script or open stack/card that has behavior**
15285	**IDE: Property inspector string value should change property when clicking outside the field.**
15247	**Default Field name should be "Field"**
15235	**Simulators not listed in Development > Test Target menu**
15227	**card 'Single Line' Message Box script doesn't pass openCard and resizestack**
15220	**Widgets Tab of extension manager is empty when reopening**
15216	**IDE: Infinite loop when resolving load order**
15180	**Can't put values of debug variables from the message box**
15001	**Extension Builder: need to set the hideConsoleWindows to true before executing shell commands**
14971	**Debugger Break Point not met on right click "Send Card/Stack message"**
14890	**Control icons missing from tools palette**
14873	**Close and remove from memory does nothing from File menu**
14852	**PI color editor doesn't react well to colors with alpha value**
14831	**Open script-only stacks in script editor when they are opened**
14822	**Edited status of stack not being set**

14738 Inspect menu missing from property inspector

14561 Widgets are not ordered in the tools palette or extension manager

Specific bug fixes (8.0.0-dp-1)

14627

13475 in the openstack handler dispatching a mouseUp to a btn does not work correctly

13447 Project Browser control layer display

13417 IDE systemVersion comparison no longer works with Yosemite

13398 Sample - Book Library.livecode edit and delete features broken

13362 Script editor opens revmenubar script when no other stack is open

13343 Cannot install Android standalone on some devices

13215 Can't type in output field of message box

13191 FIX: flip graphic horizontally and vertically for complex graphics

13159 Palettes not observing decorations under certain circumstances

12880 File->Exit should be File->Quit

11755 flip graphic gives erroneous results with complex graphics

LiveCode Builder changes

LiveCode Builder Tools

Compiler generates an error if integer literal too big

The compiler will generate an error if an integer literal is too big to fit into the (current) unsigned 32-bit integer representation.

lc-compile

Command-line interface

- A new --verbose command line flag has been added. If it is specified, **lc-compile** will output additional debugging information.

Warnings

- A new warning has been added for identifiers that may conflict with syntax keywords.
- metadata definitions that occur before module imports no longer

trigger a warning.

Bugs fixed

[14893] Emit warnings for identifiers that may cause problems.
[14939] Compiler truncates integer literals if too big.
[14950] Integer literal pattern is too general.
[14956] Do not warn for metadata before use.
[15029] Ensure that exit repeat works correctly in nested repeat up to / down to / for each loops.

LiveCode Builder Language

Case-Sensitivity

- All identifiers are now case-insensitive - i.e. a handler Main can be called as mAin, MAIN and main.

Foreign handler definitions require explicit typing.

- A foreign handler definition must declare an explicit return type.
- Each parameter in a foreign handler definition must declare an explicit type.

Replace concept of 'undefined' with 'nothing'

- The use of the keyword 'undefined' is now deprecated, 'nothing' should be used instead.
    - Use 'returns nothing' to indicate a handler which returns no value.
    - Use 'nothing' to indicate no value when manipulating optionally type variables
- The 'is defined', 'is undefined', 'is not defined', 'is not undefined' syntax is now deprecated, 'is' and 'is not' should be used with 'nothing' instead
    - Use <expr> is nothing and <expr> is not nothing to test whether an expression has a value or not
    - The phrase <left> is <right> will now return true if <left> and <right> are both nothing

    - The phrase <left> is not <right> will now return true if one of <left> or <right> are nothing (but not both).

IntSize Type

- There is now an IntSize foreign type, mapping ssize_t.

Change to handler return type syntax.

- The syntax for declaring the return type for a handler, or handler type has been changed to ['returns' 'nothing' | 'returns' <Type>].
- The previous syntax as <Type> or as undefined will continue to be supported until dp-4 at which point it will be removed along with the undefined type keyword.
- The compiler will emit warnings for the use of the deprecated syntax.

Syntax

- Syntax keywords are no longer permitted to match [A-Z0-9_.].
- metadata definitions may now occur anywhere a module's top-level

context.

- use declarations may now occur anywhere in a module's top-level

context.

Identifiers

- Identifiers are now expected to match [A-Z0-9_.].

Bugs fixed

[14906] Change 'as <Type>' to 'returns nothing' or 'returns <Type>' in handler return type definitions.
[14933] Make identifiers case-insensitive.
[14991] No 'use' clauses causes random parsing errors
[15426] Make it a compile-time error to not type foriegn handler definitions.

LiveCode Builder Host Library

Ability to access a widget's effective font

- The textFont, textSize and textStyle properties have been reserved to the host.
- New syntax my font has been added which returns a Canvas.Font matching the current effective values of the text properties that have been set on the widget.

Determining if a widget is enabled

- It is now possible to determine the enabled state of a widget from within its script.
    - The my enabled property returns true if the widget is currently enabled
    - The my disabled property returns true if the widget is currently disabled
    - If script changes the enabled (or disabled) property of the widget then an OnParentPropChanged message will be sent.

Widget Printing

- Widgets now print along with other controls.
    - Widgets will be rasterized at screen resolution and then printed as an image.
    - Higher-fidelity printing of widgets will be implemented at a later date.

Native Code Access

LiveCode extensions can now contain native code libraries which LCB will use to resolve foreign handler references.
The foreign handler binding string should be of the form libname>function to use this feature. In this case, the engine will look for a library libname on a per-platform basis when the foreign handler needs to be resolved.
Native code libraries should be present inside the resources folder inside the extension archive. The engine derives the appropriate path from the requested library name and current platform. The structure is as follows:

```
<extension>/
    resources/
        code/
            mac/
                <library>.dylib
            linux-x86/
                <library>.so
            linux-x86_64/
                <library>.so
            win-x86/
                <library>.dll
```

Note: At present, only the desktop platforms are supported.
Note: The above structure is likely to change in a future release. In particular the code folder will sit at the same level as resources rather than within it.

Ability to display a popup menu

- New syntax has been added to popup a menu constructed from a provided menu text.
    - popup menu <MenuText> at <Point>

Detecting successive clicks

- The OnClick event is sent every time a mouseDown/mouseUp sequence is detected by the engine on a widget.
- Use 'the click count' syntax to fetch the number of successive clicks which happened close together and within a certain time of each other.

Composed widgets

The ability to compose widget objects has been added. Widgets can either be 'host' widgets, created when a widget is directly embedded in a stack, or 'child' widgets which are created when a widget is used as a child widget within another widget.

Example

See
https://github.com/runrev/livecode/blob/develop/extensions/widgets/simplecomposed/simplecomposed.lcb
for an example of how the host/child relationship can be used.

Syntax

A Widget type has been added, so that variables can contain references to child widget objects.
A variable to hold a widget reference can be defined in the usual way, e.g.

variable tWidget as Widget

New widget syntax has been added to create, place, unplace and manipulate child widgets.

- a new widget <kind> - Creates a widget object of the specified kind.
- place <widget> [at (bottom|top) | (below|above) <other widget>] - Adds a child widget to the parent on the specified layer.
- unplace <widget> - Removes a child widget from the parent.
- the target - Returns the child widget that started the current execution.
- my children - Returns a list of the currently placed child widgets of this widget.
- property <property> of <widget> - Enables manipulation of a property implemented by a child widget.
- the rectangle of <widget> - Enables manipulation of the rectangle property of a child widget.
- the width of <widget> - Enables manipulation of the width property of a child widget.
- the height of <widget> - Enables manipulation of the height property of a child widget.
- the location of <widget> - Enables manipulation of the location property of a child widget.
- the enabled of <widget> - Enables manipulation of the enabled property of a child widget.
- the disabled of <widget> - Enables manipulation of the disabled property of a child widget.
- annotation <name> of <widget> - Enables tagging of child widgets with named values.

Events

Events triggered on child widgets (such as OnMouseUp) are automatically passed up to the parent, as long as the child's event handler returns nothing. If any event handler returns something, the event is considered handled and is not passed to the parent.

Messages

Messages posted by the child widget can be handled by the parent in an On<message name> handler. For example, if the child has the code

post "dataChanged" with [mDataArray],
this can be handled in the parent by adding
public handler OnDataChanged(in pArray as Array).
Posted messages can only be handled by a direct parent, and a widget's script object will only receive messages posted by host widget, i.e. the topmost parent.

Bugs fixed
[14541] Widgets should be able to popup system menus
[14805] LCB-Canvas: close path on mPath example is incorrect
[14898] Add access to widget's effective font.
[14964] Crash when calling a handler with 'any' type parameter form LCS.
[14997] image with file accepts url prefix
[15005] Logging a list fails if its element types are not strings or booleans

[15012] Fix OnClick and provide a way to get the number of successive clicks.
[15035] Svg path loses data in roundtrip
[15060] Printing a card containing a widget causes a crash.
[15115] Initialization issue with widget used in 'popup widget'
[15128] OnMouseMove doesn't fire if there is more than one widget on a card.
[15138] Map elements inside lists to arrays correctly.
[15152] Setting the alpha of one color variable changes alpha of other color variable
[15276] Keywords don't map to the correct property name.
[15284] Custom properties not saved with widgets.
[15331] Issues with char "is in" operator with non-BMP characters
[15410] Fix error with 'the local date' on Windows
[15489] 'image from resource file' syntax does not bind to anything.
[15529] Array log doesn't call string conversion recursively.
LiveCode Tools
Bugs fixed
[14938] The compiler now checks a property get handler does not return nothing.
LiveCode Builder Documentation
Bugs fixed
[15628] Typo in LCB API for next repeat
LiveCode Builder Standard Library
Sequence operations
New syntax has been added for searching partial contents of sequence types (List, String and Data) based on the offset operation.
the offset of <Needle> before <Position> in <Haystack>

- the offset of <Needle> after <Position> in <Haystack>
- Equivalent syntax has been added for the index operation.

Mathematical functions

- Several mathematical functions now throw "domain errors" when applied to values that the function is not defined for, including

log10(), ln(), asin() and acos(), and x ^ y.

Bugs fixed
[14678] Throw error on domain error in log10() and ln()
[14679] Throw error on domain error in pow()
[14681] Throw error on domain error in asin() and acos()
[14846] Fixed and reinstated "offset before" and "offset after"
[14889] Throw error when "converted from base _" fails
[15768] Fix "the local date" on Windows

Dictionary additions

- **create widget** (*command*) has been added to the dictionary.
- **load extension** (*command*) has been added to the dictionary.
- **unload extension** (*command*) has been added to the dictionary.
- **loadedExtensions** (*function*) has been added to the dictionary.
- **newWidget** (*message*) has been added to the dictionary.
- **widget** (*object*) has been added to the dictionary.
- **kind** (*property*) has been added to the dictionary.

Dictionary changes

- The entry for **accept** (*command*) has been updated.
- The entry for **close socket** (*command*) has been updated.
- The entry for **export with palette** (*command*) has been updated.
- The entry for **open socket** (*command*) has been updated.
- The entry for **read from socket** (*command*) has been updated.
- The entry for **resolve image** (*command*) has been updated.
- The entry for **secure socket** (*command*) has been updated.
- The entry for **write to socket** (*command*) has been updated.
- The entry for **clickLoc** (*function*) has been updated.
- The entry for **clickV** (*function*) has been updated.
- The entry for **commandArguments** (*function*) has been updated.
- The entry for **commandName** (*function*) has been updated.
- The entry for **mobileGetLaunchData** (*function*) has been updated.
- The entry for **mobileStorePurchaseError** (*function*) has been updated.
- The entry for **mouseH** (*function*) has been updated.
- The entry for **mouseV** (*function*) has been updated.
- The entry for **openSockets** (*function*) has been updated.
- The entry for **specialFolderPath** (*function*) has been updated.
- The entry for **codepoint** (*keyword*) has been updated.
- The entry for **codepoints** (*keyword*) has been updated.
- The entry for **dateItems** (*keyword*) has been updated.
- The entry for **trueWord** (*keyword*) has been updated.
- The entry for **trueWords** (*keyword*) has been updated.
- The entry for **purchaseStateUpdate** (*message*) has been updated.
- The entry for **remoteControlReceived** (*function*) has been updated.
- The entry for **socketClosed** (*message*) has been updated.
- The entry for **socketError** (*message*) has been updated.
- The entry for **socketTimeout** (*message*) has been updated.
- The entry for **HTMLText** (*property*) has been updated.
- The entry for **activatePalettes** (*property*) has been updated.
- The entry for **clipboardData** (*property*) has been updated.
- The entry for **innerGlow** (*property*) has been updated.
- The entry for **password** (*property*) has been updated.
- The entry for **serialControlString** (*property*) has been updated.
- The entry for **socketTimeoutInterval** (*property*) has been updated.
- The entry for **sslCertificates** (*property*) has been updated.
- The entry for **stackFiles** (*property*) has been updated.

Previous Release Notes

7.0.6 Release Notes	http://downloads.livecode.com/livecode/7_0_6/LiveCodeNotes-7_0_6.pdf
7.0.4 Release Notes	http://downloads.livecode.com/livecode/7_0_4/LiveCodeNotes-7_0_4.pdf
7.0.3 Release Notes	http://downloads.livecode.com/livecode/7_0_3/LiveCodeNotes-7_0_3.pdf
7.0.1 Release Notes	http://downloads.livecode.com/livecode/7_0_1/LiveCodeNotes-7_0_1.pdf
7.0.0 Release Notes	http://downloads.livecode.com/livecode/7_0_0/LiveCodeNotes-7_0_0.pdf
6.7.6 Release Notes	http://downloads.livecode.com/livecode/6_7_6/LiveCodeNotes-6_7_6.pdf
6.7.4 Release Notes	http://downloads.livecode.com/livecode/6_7_4/LiveCodeNotes-6_7_4.pdf
6.7.2 Release Notes	http://downloads.livecode.com/livecode/6_7_2/LiveCodeNotes-6_7_2.pdf
6.7.1 Release Notes	http://downloads.livecode.com/livecode/6_7_1/LiveCodeNotes-6_7_1.pdf
6.7.0 Release Notes	http://downloads.livecode.com/livecode/6_7_0/LiveCodeNotes-6_7_0.pdf
6.6.2 Release Notes	http://downloads.livecode.com/livecode/6_6_2/LiveCodeNotes-6_6_2.pdf
6.6.1 Release Notes	http://downloads.livecode.com/livecode/6_6_1/LiveCodeNotes-6_6_1.pdf
6.6.0 Release Notes	http://downloads.livecode.com/livecode/6_6_0/LiveCodeNotes-6_6_0.pdf
6.5.2 Release Notes	http://downloads.livecode.com/livecode/6_5_2/LiveCodeNotes-6_5_2.pdf
6.5.1 Release Notes	http://downloads.livecode.com/livecode/6_5_1/LiveCodeNotes-6_5_1.pdf
6.5.0 Release Notes	http://downloads.livecode.com/livecode/6_5_0/LiveCodeNotes-6_5_0.pdf
6.1.3 Release Notes	http://downloads.livecode.com/livecode/6_1_3/LiveCodeNotes-6_1_3.pdf
6.1.2 Release Notes	http://downloads.livecode.com/livecode/6_1_2/LiveCodeNotes-6_1_2.pdf
6.1.1 Release Notes	http://downloads.livecode.com/livecode/6_1_1/LiveCodeNotes-6_1_1.pdf
6.1.0 Release Notes	http://downloads.livecode.com/livecode/6_1_0/LiveCodeNotes-6_1_0.pdf