LiveCode 7.0.0 Release Notes

Table of contents

Overview

The LiveCode engine has undergone a large quantity of changes for the 7.0 release. The way values of variables are stored internally has been changed - in particular where before the engine used C-strings, it now uses a reference counted MCStringRef type. Every bit of code that displays text in LiveCode has been updated, and all the platform-specific API functions that manipulate characters now use the Unicode versions; as a result LiveCode is now fully Unicode compatible.

The implementation of Unicode compatibility necessitated a change to the stack file format, which means stacks saved in 7.0 format are not compatible with earlier versions of LiveCode. However you can still save stacks in legacy formats using the dropdown menu in the Save As... dialog.

The other significant change to engine internals is the work done on syntax refactoring. The code that deals with statement execution, function evaluation and property access has been cleaned up and separated out from the parsing code, and moved into distinct modules based on functionality. This represents a major first step towards being able to implement Open Language.

Known issues

Every effort has been made to ensure that externally, the engine behaviour is identical to the current unrefactored release. In other words, users should not notice any difference in functionality in their existing stacks. However, users will notice a general slow-down caused by lack of optimisation in this release - this will be addressed for DP 2.

- 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 engine files are much larger than previous versions due to inclusion of ICU data
- LiveCode does not run correctly when installed to Unicode paths on OSX
- On Windows, executing LiveCode from the installer fails as it cannot find the IDE
- Android app label is not yet Unicode compatible
- Auto-updater process doesn't terminate when dismissed

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 7.0.0
- Linux: /opt/runrev/livecode-7.0.0
- Mac: /Applications/ LiveCode 7.0.0.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 7.0.0
- Linux: ~/.runrev/components/livecode-7.0.0
- Mac: ~/Applications/ LiveCode 7.0.0.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-7.0.0
```

- 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.

Proposed changes

The following changes are likely to occur in the next or subsequent non-maintenance release:

- The engine (both IDE and standalone) **will require** gtk, gdk and glib on Linux

Engine changes

Cocoa Support (7.0.0)

With 6.7 we have replaced the majority of Carbon API usage with Cocoa. The goals of this work are three-fold:

- Allow embedding of native 'NSViews' into LiveCode windows (in particular, browser controls).
- Enable submission of LiveCode apps to the Mac AppStore.
- Enable eventual building of 64-bit versions of LiveCode for Mac.

We have achieved the first two of these goals in 6.7.

The instability issues caused by the AppStore sandbox when using mixed Cocoa and Carbon APIs has been resolved - LiveCode apps built with 6.7 can be successfully sandboxed and thus submitted to the AppStore.

The dontUseQT property is now true by default on Mac. This means that, by default, the AVKit implementation of the player will be used on 10.8 and above. Note that, as it stands, when dontUseQT is true neither QT visual effects nor sound recording will work.

The final goal (64-bit support) will be gradually worked towards over the next few LiveCode versions as the engine gets 'decarbonated' (usage of Carbon APIs which do not have 64-bit equivalents removed).

An important internal change which will affect maintainers of Mac externals that use the windowId is that this property now returns the 'global window number' (which is the unique ID the Window Server uses to identify windows). To turn this into a Cocoa NSWindow pointer use [NSApp windowWithWindowNumber: t_window_id]. Note that it is no longer possible to get a Carbon WindowRef, nor should this be attempted as trying to mix Carbon and Cocoa in this manner will cause instability inside the sandbox environment required by the Mac AppStore.

An important script visible change that has occurred due to the move to Cocoa is screen updating. Previously (when using Carbon) the OS would 'coalesce' successive requests to update the screen - the window buffer would be updated, but the window buffer would only be flushed when the OS decided to. In Cocoa, after a screen update the window buffer is *always* flushed. Outside of 'lock screen', the engine applies any screen updates after each command execution therefore in 6.7+ make sure you use lock screen around blocks of code that make many screen updates - unless you want each update to be visible. It should be noted that the behavior in 6.7 is now the same as on Windows and Linux however the OS takes longer to flush window updates to the screen on Mac than on the other platforms meaning that using lock screen is important.

Note: QTVR movies are no longer supported as they are not supported by QTKit nor AVKit.

Note: Drawers no longer work on Mac, they will appear as normal stacks.

Fix UTF-8 output from server scripts (7.0.0)

Export "the styledText" runs as "text" rather than "unicodeText" (7.0.0)

Fix multiple middle-click pasting issues (7.0.0)

"is an array" is only true if there is at least one key (7.0.0)

charToNum(empty) should return empty (7.0.0)

Always insert a linebreak after vtab (7.0.0)

Various fixes to binaryDecode (7.0.0)

Strings should convert to empty arrays (7.0.0)

Location Services Disabled with LC 6.6.4 (rc1) (7.0.0)

A new function **mobileLocationAuthorizationStatus** (or **iphoneLocationAuthorizationStatus**) has been added. This returns the current location authorization status of the calling application. The status can be one of the following:

- **notDetermined**: User has not yet made a choice with regards to this application
- **restricted**: The application is not authorized to use location service
- **denied**: User has explicitly denied authorization for this application, or location services are disabled in Settings.
- **authorizedAlways**: User has granted authorization to use their location at any time, including monitoring for regions, visits, or significant location changes.
- **authorizedWhenInUse**: User has granted authorization to use their location only when the app is visible to them (it will be made visible to them if you continue to receive location updates while in the background). Authorization to use launch APIs has not been granted.

We have also changed the flow of the messages being sent to the user when using Location Services in iOS 8:

- In the standalone application settings tab, the developer can choose the type of the authorization request for their app.
The two available options are either "always" or "when in use". Selecting "always" means that the app will prompt the user to grant authorization to use their location
at *any* time, including monitoring for regions, visits, or significant location changes. The app then has access to the user's location even when the app is in the
background. On the contrary, if "when in use" is selected, the app will prompt the user to grant authorization to use their location only when the app is visible on screen. You can choose only one type, not both. This means that if you go to Settings -> Privacy -> Location, you will see only two choices available ("Never" and either "Always" or "While using the app") for this app, keeping it consistent with other iOS apps.

- When the app is installed (on device or simulator) for the very first time, a dialog will pop up asking the user to authorize the app to use their location
"always" or "when in use", depending on what was previously chosen in the standalone application settings.

- Every time the app is launched, it remembers the user's preference. No other popup dialogs will appear.

- The user can at any time change their preferences in Settings -> Privacy -> Location -> ..

- In that way, you need not modify your existing scripts that used Location Services, in order to add iOS 8 support.

Multimedia on MacOS with AVFoundation (7.0.0-rc-3)

What has changed?

The player object until now used QuickTime/QTKit APIs for audio and video playback. Since both

QuickTime and QTKit have been deprecated by Apple, we have updated the player to use the new AVFoundation API. AVFoundation does not provide a controller for multimedia playback until OSX 10.9 and their new control bar is also missing some of the features provided by the QTKIt controller, which required us to implement our own controller to ensure backward compatibility.
We have added two new properties to the player object enabling you to customise the appearance of the controller:

- The **hilitecolor** of a player is the color of the played area, the colour of the volume area, as well as the background color of a controller button when it is pressed.

- The **forecolor** of a player is the color of the selected area. The selected area is the area between the selection handles.

We have also added support for getting information about the download progress of a remote multimedia file:

- The **loadedtime** of a player is the time up to which the movie can be played. The download progress is also displayed on the controller well.

You can also query the **status** property of the player. This property can take either of the values:
- **loading** (for remote multimedia files)
- **playing**
- **paused**

A new message is added to the player:
- The **playRateChanged** message is sent to the player when the rate is changed by the rate scrollbar controller. To enable the rate scrollbar controller, hold shift + click on scrubForward/scrubBack buttons of the player controller.

Note AVFoundation player is supported in OSX 10.8 and above. On systems running OSX 10.6 and 10.7, LiveCode continues to provide player functionality using the QTKit API.

BidiDirection (7.0.0-rc-3)

The bidiDirection is a function that has been added to expose the engine's implementation of the Unicode Bidirectional Algorithm. It returns "ltr" or "rtl", depending on the computed base direction of the text it receives as a parameter.

iOS 8 Support (7.0.0-rc-3)

Support for iOS 8 device and simulator builds has been added to 6.6.4-rc-1 for OS 10.9 users. This means that if you are using OS 10.9 you must now have Xcode 6 installed in order to perform device builds. The requirements for all previous OS X versions will remain the same.

Bugs relating to orientation, push notifications and screen sizes on iOS 8 have been resolved in addition to standalone builder updates allowing for the specification of new iPhone 6 splash screens.

Enable "umask" property on OS X (7.0.0-rc-3)

On POSIX systems, it is sometimes useful to set the umask when creating files or directories. For example, this can be useful when creating temporary directories.

Previously, the "umask" property in LiveCode was only implemented on iOS, Linux and Android platforms. It is now also available on Mac OS X.

Fix a bug in the image saving code causing stackfile corruption (7.0.0-rc-3)

Copy files do not work with the iOS 8 simulator (7.0.0-rc-3)

This fix has been tweaked for 6.6.4-rc-3. If, when attempting to deploy to the iOS 8 simulator you get the error "Unable to start simulation: Unable to run app in Simulator", delete any previous version of the app installed on the simulator and redeploy.

Fix calculation for tab-on-return (7.0.0-rc-3)

Fix string -> bool conversion in the v1 externals interface (7.0.0-rc-3)

Fix OSX specialFolderPath("asup") (7.0.0-rc-3)

Fix a typo in the Win32 time formatting code (7.0.0-rc-3)

Fix "answer file" opening in wrong folder (7.0.0-rc-3)

Fix a crash due to uninitialised locale on server (7.0.0-rc-3)

Fix a potential nil pointer crash (7.0.0-rc-3)

Mark the installer as retina-capable (7.0.0-rc-3)

Linux: update engine mouse coords on click events (7.0.0-rc-3)

Fix post-install launching on Linux (7.0.0-rc-3)

Fix I/O for serial devices (7.0.0-rc-3)

Fix Windows command line parsing (7.0.0-rc-3)

Use correct pixel order for OSX PPC (7.0.0-rc-3)

File format change (7.0.0-rc-2)

In order to accommodate the saving and loading of unicode text throughout LiveCode, the file format of stacks has been changed. This means that stacks saved in 7.0 format cannot be opened in previous versions of LiveCode.

Legacy file formats are available to select when using the Save As... dialog. Saving in a legacy format will result in the loss of some information related to LiveCode 7.0, namely Unicode text in some areas (for example in object scripts), right-to-left formatting and tab alignment.

Array element pass by reference (7.0.0-rc-2)

It is now possible to pass parts of an array by reference. For example, the following

```
on mouseUp
```

```
   local tArray

   put "" into tArray[1][2]

   passByRef tArray[1]

   put tArray[1][2]

end mouseUp



on passByRef @rArray

   put "changed" into rArray[2]

end passByRef
```

in the script of a button will result in "changed" appearing in the message box when the button is pressed.

This allows users to reduce the overhead associated with passing sub-arrays to handlers, as this would no longer require copying the sub-array internally.

Fix deployment to Windows from 64-bit Linux (7.0.0-rc-2)

Fix a null-pointer crash on Linux server (7.0.0-rc-2)

"the processor" returns "arm" on RaspberryPi (7.0.0-rc-2)

Fix system time formatting on Windows (7.0.0-rc-2)

Hebrew text is shown in reverse character order on Android (7.0.0-dp-7)

This bug fix involved incorporating the HarfBuzz library in Android builds. In addition to resolving bugs related to RTL text display, this has also enabled support for complex text shaping, so that combinations of characters in complex scripts such as Arabic are displayed correctly.

Unicode Support (7.0.0-dp-1)

Unicode and LiveCode

Traditionally, computer systems have stored text as 8-bit bytes, with each byte representing a single character (for example, the letter 'A' might be stored as 65). This has the advantage of being very simple and space efficient whilst providing enough (256) different values to represent all the symbols that might be provided on a typewriter.

The flaw in this scheme becomes obvious fairly quickly: there are far more than 256 different characters in use in all the writing systems of the world, especially when East Asian ideographic languages are considered. But, in the pre-internet days, this was not a big problem.

LiveCode, as a product first created before the rise of the internet, also adopted the 8-bit character sets of the platforms it ran on (which also meant that each platform used a different character set: MacRoman on Apple devices, CP1252 on Windows and ISO-8859-1 on Linux and Solaris). LiveCode terms these character encodings "native" encodings.

In order to overcome the limitations of 8-bit character sets, the Unicode Consortium was formed. This group aims to assign a unique numerical value ("codepoint") to each symbol used in every written language in use (and in a number that are no longer used!). Unfortunately, this means that a single byte cannot represent any possible character.

The solution to this is to use multiple bytes to encode Unicode characters and there are a number of schemes for doing so. Some of these schemes can be quite complex, requiring a varying number of bytes for each character, depending on its codepoint.

LiveCode previously added support for the UTF-16 encoding for text stored in fields but this could be cumbersome to manipulate as the variable-length aspects of it were not handled transparently and it could only be used in limited contexts. Unicode could not be used in control names, directly in scripts or in many other places where it might be useful.

In LiveCode 7.0, the engine has been extensively re-written to be able to handle Unicode text transparently throughout. The standard text manipulation operations work on Unicode text without any additional effort on your part; Unicode text can now be used to name controls, stacks and other objects; menus containing Unicode selections no longer require tags to be usable - anywhere text is used, Unicode should work.

Adding this support has required some changes but these should be minor. Existing apps should continue to run with no changes but some tweaking may be required in order to adapt them for full Unicode support - this is described in the next section - Creating Unicode Apps.

Creating Unicode Apps

Creating stacks that support Unicode is no more difficult than creating any other stack but there are a few things that should be borne in mind when developing with Unicode. The most important of these is the difference between text and binary data - in previous versions of LiveCode, these could be used interchangeably; doing this with Unicode may not work as you expect (but it will continue to work for non-Unicode text).

When text is treated as binary data (i.e when it is written to a file, process, socket or other object outside of the LiveCode engine) it will lose its Unicode-ness: it will automatically be converted into the platform's 8-bit native character set and any Unicode characters that cannot be correctly represented will be converted into question mark '?' characters.

Similarly, treating binary data as text will interpret it as native text and won't support Unicode.

To avoid this loss of data, text should be explicitly encoded into binary data and decoded from binary data at these boundaries - this is done using the **textEncode** and **textDecode** functions (or its equivalents, such as opening a file using a specific encoding).

Unfortunately, the correct text encoding depends on the other programs that will be processing your data and cannot be automatically detected by the LiveCode engine. If in doubt, UTF-8 is often a good choice as it is widely supported by a number of text processing tools and is sometimes considered to be the "default" Unicode encoding.

New & Existing apps - things to look out for

- When dealing with binary data, you should use the **byte** chunk expression rather than **char** - **char** is

intended for use with textual data and represents a single graphical character rather than an 8-bit unit.
- Try to avoid hard-coding assumptions based on your native language - the formatting of numbers or the correct direction for text layout, for example. LiveCode provides utilities to assist you with this.
- Regardless of visual direction, text in LiveCode is always in logical order - word 1 is always the first word; it does not depend on whether it appears at the left or the right.
- Even English text can contain Unicode characters - curly quotation marks, long and short dashes, accents on loanwords, currency symbols...

New Commands, Functions & Syntax

Chunk expressions: byte, char, codepoint, codeunit

byte *x* **to** *y* **of** *text* -- Returns bytes from a binary string
char *x* **to** *y* **of** *text* -- As a series of graphical units
codepoint *x* **to** *y* **of** *text* -- As a series of Unicode codepoints
codeunit *x* **to** *y* **of** *text* -- As a series of encoded units

A variety of new chunk types have been added to the LiveCode syntax to support the various methods of referring to the components of text. This set is only important to those implementing low-level functions and can be safely ignored by the majority of users.

The key change is that **byte** and **char** are no longer synonyms - a byte is strictly an 8-bit unit and can only be reliably used with binary data. For backwards compatibility, it returns the corresponding native character from Unicode text (or a '?' if not representable) but this behaviour is deprecated and should not be used in new code.

The **char** chunk type no longer means an 8-bit unit but instead refers to what would naturally be thought of as a single graphical character (even if it is composed of multiple sub-units, as in some accented text or Korean ideographs). Because of this change, it is inappropriate to use this type of chunk expression on binary data.

The **codepoint** chunk type allows access to the sequence of Unicode codepoints which make up the string. This allows direct access to the components that make up a character. For example, á can be encoded as (a,combining-acute-accent) so it is one character, but two codepoints (the two codepoints being a and combining-acute-accent).

The **codeunit** chunk type allows direct access to the UTF-16 code-units which notionally make up the internal storage of strings. The codeunit and codepoint chunk are the same if a string only contains unicode codepoints from the Basic Multilingual Plane. If, however, the string contains unicode codepoints from the Supplementary Planes, then such codepoints are represented as two codeunits (via the surrogate pair mechanism). The most important feature of the 'codeunit' chunk is that it guarantees constant time indexed access into a string (just as char did in previous engines) however it is not of general utility and should be reserved for use in scripts which need greater speed but do not need to process Supplmentary Plane characters, or are able to do such processing themselves.

The hierarchy of these new and altered chunk types is as follows: **byte** *w* of **codeunit** *x* of **codepoint** *y* of **char** *z* of **word**...

Chunk expressions: paragraph, sentence and trueWord

The **sentence** and **trueWord** chunk expressions have been added to facilitate the processing of text, taking into account the different character sets and conventions used by various languages. They use the ICU library, which uses a large database of rules for its boundary analysis, to determine sentence and word breaks. ICU word breaks delimit not only whitespace but also individual punctuation characters; as a result the LiveCode **trueWord** chunk disregards any such substrings that contain no alphabetic or numeric characters.

The **paragraph** chunk is identical to the existing **line** chunk, except that it is also delimited by the Unicode paragraph separator (0x2029), which reflects paragraph breaking in LiveCode fields.

The hierarchy of these new chunk types is as follows: **trueword** *v* of **word** *w* of **item** *x* of **sentence** *y* of **paragraph** *z* of **line**...

Synonym: segment

The **segment** chunk type has been added as a synonym to the existing **word** chunk. This in order to allow you to update your scripts to use the newer syntax in anticipation of a future change to make the behaviour of the **word** chunk match the new **trueWord** behaviour.

We would anticipate changing the meaning of **word** with our 'Open Language' project. It requires us to create a highly accurate script translation system to allow old scripts to be rewritten in new revised and cleaner syntax. It is at this point we can seriously think about changing the meaning of existing tokens, including **word**. Existing scripts will continue to run using the existing parser, and they can be converted (by the user) over time to use the newer syntax.

Property: the formSensitive

set the **formSensitive** to false -- Default value

This property is similar to the **caseSensitive** property in its behaviour - it controls how text with minor differences is treated in comparison operations.

Normalization is a process defined by the Unicode standard for removing minor encoding differences for a small set of characters and is more fully described in the **normalizeText** function.

Command: open file/process/socket ... for <encoding> text

open file *"log.txt"* **for utf-8 text read** -- Opens a file as UTF-8

Opens a file, process or socket for text I/O using the specified encoding. The encodings supported by this command are the same as those for the **textEncode** / **textDecode** functions. All text written to or read from the object will undergo the appropriate encoding/decoding operation automatically.

Functions: textEncode, textDecode

textEncode(*string*, *encoding*) -- Converts from text to binary data
textDecode(*binary*, *encoding*) -- Converts from binary data to text

Supported encodings are (currently):

- "ASCII"
- "ISO-8859-1" (Linux only)
- "MacRoman" (OSX only)
- "Native" (ISO-8859-1 on Linux, MacRoman on OSX, CP1252 Windows)
- "UTF-16"
- "UTF-16BE"
- "UTF-16LE"
- "UTF-32"
- "UTF-32BE"

- "UTF-32LE"
- "UTF-8"
- "CP1252" (Windows only)

Spelling variations are ignored when matching encoding strings (i.e all characters other than [a-zA-z0-9] are ignored in matches as are case differences).

It is very highly recommended that any time you interface with things outside LiveCode (files, network sockets, processes, etc) that you explicitly **textEncode** any text you send outside LiveCode and **textDecode** all text received into LiveCode. If this doesn't happen, a platform-dependent encoding will be used (which normally does not support Unicode text).

It is not, in general, possible to reliably auto-detect text encodings so please check the documentation for the programme you are communicating with to find out what it expects. If in doubt, try "UTF-8".

Functions: numToCodepoint, codepointToNum

numToCodepoint(*number*) -- Converts a Unicode codepoint to text
codepointToNum(*codepoint*) -- Converts a codepoint to an integer

These functions convert between the textual form of a Unicode character and its numerical identifier ("codepoint"). Codepoints are integers in the range 0x000000 to 0x10FFFF that identify Unicode characters. For example, the space (" ") character is 0x20 and "A" is 0x41.

The codepointToNum function raises an exception if the argument contains multiple codepoints; it should generally be used in the form:

```
codepointToNum(codepoint x of string)
```

The numToCodepoint function raises an exception if the given integer is out of range for Unicode codepoints (i.e if it is negative or if it is greater than 0x10FFFF). Codepoints that are not currently assigned to characters by the latest Unicode standard are not considered to be invalid in order to ensure compatibility with future standards.

Functions: numToNativeChar, nativeCharToNum

numToNativeChar(*number*) -- Converts an 8-bit value to text
nativeCharToNum(*character*) -- Converts a character to an 8-bit value

These functions convert between text and native characters and are replacements for the deprecated **numToChar** and **charToNum** functions.

As the "native" character sets for each platform have a limited and different repertoire, these functions should not be used when preservation of Unicode text is desired. Any characters that cannot be mapped to the native character set are replaced with a question mark character ('?').

Unless needed for compatibility reasons, it is recommended that you use the **numToCodepoint** and **codepointToNum** functions instead.

Function: normalizeText

normalizeText(*text*, *normalForm*) -- Normalizes to the given form

The **normalizeText** function converts a text string into a specific 'normal form'.

Use the **normalizeText** function when you require a specific normal form of text.

In Unicode text, the same visual string can be represented by different character sequences. A prime example of this is precomposed characters and decomposed characters: an 'e' followed by a combining acute character is visually indistinguishable from a precombined 'é' character. Because of the confusion that can result, Unicode defined a number of "normal forms" that ensure that character representations are consistent.

The normal forms supported by this function are:

- "NFC" - precomposed
- "NFD" - decomposed
- "NFKC" - compatibility precomposed
- "NFKD" - compatibility decomposed

The "compatibility" normal forms are designed by the Unicode Consortium for dealing with certain legacy encodings and are not generally useful otherwise.

It should be noted that normalization does not avoid all problems with visually-identical characters; Unicode contains a number of characters that will (in the majority of fonts) be indistinguishable but are nonetheless completely different characters (a prime example of this is "M" and U+2164 "M" ROMAN NUMERAL ONE THOUSAND).

Unless the **formSensitive** handler property is set to true, LiveCode ignores text normalization when performing comparisons (is, <>, etc).

Returns: the text normalized into the given form.

```
set the formSensitive to true

put "e" & numToCodepoint("0x301") into tExample  -- Acute accent

put tExample is "é"      -- Returns false

put normalizeText(tExample, "NFC") is "é"  -- Returns true
```

Function: codepointProperty

codepointProperty("A", "Script") -- "Latin"
codepointProperty("β", "Uppercase") -- false
codepointProperty("σ", "Name") -- GREEK SMALL LETTER SIGMA

Retrieves a UCD character property of a Unicode codepoint.

The Unicode standard and the associated Unicode Character Database (UCD) define a series of properties for each codepoint in the Unicode standard. A number of these properties are used internally by the engine during text processing but it is also possible to query these properties directly using this function.

This function is not intended for general-purpose use; please use functions such as toUpper or the "is" operators instead.

There are many properties available; please see the version 6.3.0 of the Unicode standard, Chapter 4 and Section 5 of Unicode Technical Report (TR)#44 for details on the names and values of properties. Property names may be specified with either spaces or underscores and are not case-sensitive.

Examples of supported properties are:

- "Name" - Unique name for this codepoint
- "Numeric_Value" - Numerical value, e.g. 4 for "4"
- "Quotation_Mark" - True if the codepoint is a quotation mark
- "Uppercase_Mapping" - Uppercase equivalent of the character
- "Lowercase" - True if the codepoint is lower-case

Updated Functions

Function: binaryEncode

A new letter has been introduced to allow one to binary encode unicode strings.
Following the dictionary definitions, it consists of:

u{<encoding>}: convert the input string to the encoding specified in the curly braces, and output up to amount bytes of the string created - stopping at the last encoded character fitting in the amount - padding with '\0'.

U{<encoding>}: convert the input string to the encoding specified in the curly braces, and output up to amount bytes of the string created - stopping at the last encoded character fitting in the amount - padding with encoded spaces, and then '\0' if the last encoded space cannot fit within the amount specified.

The encoding, surrounded by curly braces, is optional - no one specified would default to the behaviour of 'a' - and must match one of those applicable to textEncode

Function: binaryDecode

A new letter has been introduced to allow one to binary decode unicode strings.
Following the dictionary definitions, it consists of:

u{<encoding>}: convert amount bytes of the input string to the specified encoding, padding with '\0'.

U{<encoding>}: converts amount bytes of the input to the specified encoding, skipping trailing spaces.

The encoding, surrounded by curly braces, is optional - no one specified would default to the behaviour of 'a' - and must match one of those applicable to textEncode

Deprecated Features

Functions: numToChar, charToNum

These functions should not be used in new code as they cannot correctly handle Unicode text.

Property: useUnicode

This property should not be used in new code, as it only affects the behaviour of **numToChar** and **charToNum**, which are themselves deprecated.

Functions: uniEncode, uniDecode

These functions should not be used in new code as their existing behaviour is incompatible with the new, transparent Unicode handling (the resulting value will be treated as binary data rather than text). These functions are only useful in combination with the also-deprecated unicode properties described below.

Function: measureUnicodeText

This function should not be used in new code. **measureUnicodeText**(*tText*) is equivalent to **measureText**(**textDecode**(*tText*, "UTF16")).

Properties: unicodeText, unicodeLabel, unicodeTitle, unicodeTooltip, unicodePlainText, unicodeFormattedText

These properties should not be used in new code; simply set the text, label, title etc. as normal. Assigning values other than those returned from **uniEncode** to these properties will not produce the desired results.

The following are now equivalent:

```
set the unicodeText of field 1 to tText

set the text of field 1 to textDecode(tText, "UTF16")
```

and similarly for the other unicode-prefixed properties.

Specific bug fixes (7.0.0)

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

13763	**Native chars don't hash to the same value as equivalent unicode chars**
13761	**Fix UTF-8 output from server scripts**
13757	**The detailed files is wrong on Windows.**
13753	**Project Browser reports incorrect control for behavior of a card**
13752	**Double-Clicking On a Player Doesn't Show Inspector**
13750	**Picker broken on iPhone 4 iOS 7.1**
13746	**the shape property of stacks is broken**
13745	**answer file with type treats empty filter as wild**
13742	**Export "the styledText" runs as "text" rather than "unicodeText"**
13741	**Fix multiple middle-click pasting issues**
13740	**numToByte outputs text rather than data**
13738	**audioClip references not being resolved correctly**
13737	**"is an array" is only true if there is at least one key**
13736	**charToNum(empty) should return empty**
13732	**Saving a stack with a binary string in a custom property in 5.5 format causes truncation at 65535 bytes.**
13728	**Issue with externals and reading values from LiveCode variables**
13727	**Always insert a linebreak after vtab**
13725	**Various fixes to binaryDecode**
13724	**Strings should convert to empty arrays**
13721	**Externals using 'LCObjectPost' don't always cause the action to trigger - particularly on Yosemite.**
13717	**Link Colors Inconsistent**

13711	**Player plays audio but not video**
13710	**[[Player]] video image not shown under some circumstances**
13708	**mobilepickphoto in landscape orientation causes an orientation change**
13707	**[[iOS 8]] Denying access to location services when the app is launched for the very first time causes the app to freeze**
13699	**iOS 8 Keyboard is invisible if privacy set to "While Using the App"**
13684	**hidePalettes property defaults to false**
13677	**iOS Picker appears under the keyboard on iOS 8**
13675	**Scrollbar for the font selection in the script editor preferences doesn't work**
13665	**Ask/answer calls in (pre)openstack cause iOS 8 apps to hang**
13658	**Data corrupted by the shell() function on server**
13622	**Make sure PATH variable passes through to shell() properly on Yosemite.**
13590	**Location Services Disabled with LC 6.6.4 (rc1)**
13510	**Shutdownrequest message sent twice when triggered from quit in menu or Cmd-Q on Mac.**
13493	**Scroll is being reset in 6.7 when it is not in 6.5.2**
13450	**Independence resolution does not work well with a Browser Object**
13351	**printing a field with listbehaviour set to true makes gray background**

Specific bug fixes (7.0.0-rc-3)

13706	Fix a bug in the image saving code causing stackfile corruption
13680	item delimiter not deleted in target string if longer than 1 character
13674	Implement diskSpace function on Linux.
13671	repeat for each item or line does not use multichar delimiter
13664	Livecode 7.0 rc 2 does not sort certain strings correctly when the numeric form is used
13662	OS X standalone can't be run out of app bundle
13660	Crash (SIGSEGV) during drag & drop operation
13659	When Voice Over is turned on
13656	iOS 8 ask and answer dialogs do not handle rotation correctly
13650	Crash when opening stack
13644	wait loop not being broken
13642	Improved documentation for the umask property.
13639	mobilepickcontact works under ios 7 but not under ios 8
13634	screenshots taken in landscape view are rotated by 90 degrees on iOS 8
13626	Android app crashes when back button pressed for a second time
13621	mobileFindContact fails silently on iOS 7.1 and higher
13619	Setting a non-readable default folder makes 'the folders' fail
13610	dragImage with id upper than 65535
13605	'set the clipboardData' can cause crashing on Windows
13594	Evaluation faulty
13587	Stacks with Unicode filename won't open from the Dock
13584	Simulator launches with incorrect version
13583	Copy files do not work with the iOS 8 simulator
13579	Behaviour for 'there is a url...' in LiveCode 7.0 inconsistent with previous versions
13569	changes to [[Player]] in preOpenCard are visible to user
13568	Extra undo of paint tools crashes live code

13559	firstindent can not be set via styledText array in 7.0 RC2
13555	keydown event not getting passed to mainstack in modal dialogs
13553	Crash when putting one image into another
13552	Crash navigating to a card
13550	Deleting word chunk erroneously removes preceding whitespace
13548	Fix calculation for tab-on-return
13542	A card with many (>80) text fields causes a crash on android device when made visible
13540	[[Player]] Shift + click in controller sets showSelection to true
13539	menuPick not triggered under certain conditions
13535	Threaded rendering crash
13534	Fix string -> bool conversion in the v1 externals interface
13530	revXMLCreateTreeFromFile does not work with decomposed accented characters in filename
13529	Setting TabWidth on a lines of a field crashes with runtime error
13528	Parentheses appear in disabled tab menu items
13526	Stack location reported incorrectly if mouse released while dragging window
13523	Fix OSX specialFolderPath("asup")
13522	pull down menus do not work properly in modal dialogs
13516	if an error is encountered after a drag-and-drop
13512	4 inch iPhone apps do not use the full screen
13511	Fix a typo in the Win32 time formatting code
13509	Livecode 7 remembers cleared block attributes
13503	PDF printing does not work correctly on iOS 8.
13501	Referenced image fails to load in 6.7.0 RC2 and 7.0 RC1
13499	Fix "answer file" opening in wrong folder
13496	Fix a crash due to uninitialised locale on server
13485	Manifest file not needed in standalone bundle
13484	mobilePick and mobilePickDate do not work in iOS 8
13480	Fix a potential nil pointer crash
13462	revPrintField clips document under some circumstances
13451	RGB imageData values (charToNum) are different on Mac / Windows
13426	mobileDeviceOrientation() not working in LC 7.0.0(rc1)
13360	LiveCode application takes up 98% of processor
13349	Go stack in window displays new stack before before preopenStack/preopenCard messages are triggered
13317	Mark the installer as retina-capable
13236	mobilePickPhoto camera view is rotated on iPad when in landscape or in portraitUpsideDown
13225	Linux: update engine mouse coords on click events
13208	Image file color profiles don't seem to be handled correctly
12876	Fix post-install launching on Linux
12786	\+\ key combination now works on Linux desktop.
12545	Fix I/O for serial devices
12464	The effective screenrect returns incorrect values when hiding/showing keyboard on iOS
12444	Fix Windows command line parsing
12142	mobileSensorReading("location"
11968	Use correct pixel order for OSX PPC

11817	major speed degradation between 7.x

Specific bug fixes (7.0.0-rc-2)

13473	Add image area to card and in the property inspector put a URL as the source
13467	PrintPageNumber returns -1 by default
13465	More memory leaks in handler parameter creation
13463	Fix deployment to Windows from 64-bit Linux
13461	sort removes textColor in 7.0 RC1
13460	'convert' output is incorrectly formatted
13454	Memory leaks in handler parameter creation
13453	variable watch not working
13444	Make sure data is sent when doing POST or PUT from LiveCode Server.
13437	Setting the currentTime of player result in error statement "Not a number" in LC 7.0
13433	No mention of file format change in v7 Release Notes
13430	Nudging an object with arrow keys is broken
13428	Play stop not working
13422	setting iconGravity needs redraw
13407	Hilite artifact on the last column in VGrid mode
13405	IDE Menu Shortcuts only work when menu is open
13403	Text wraps when field width set to formattedWidth
13401	Setting a button label to empty when it has one already does not reinstate button name as label
13400	Severe slowdown in copying blocks of bytes / chars
13394	go next marked cd does not work anymore
13388	Put text after a buttons text causes LC7 RC1 to crash
13385	Fix a null-pointer crash on Linux server
13378	'get' causes a crash on server
13375	setting tabWidths results in erroneous tabstops in LC7RC1
13361	Script editor replaces *more* than the selection with new text
13359	matchText can't assign values to variables that are parameters
13356	empty not among the items of a list with a trailing comma
13353	Getting htmltext when there is firstindent results in LiveCode crashing
13352	Crash when sorting lines of a field
13348	RawKeyUp and KeyUp are sent twice in LC 7
13346	LineOffset should return 0 in LC7 RC1
13340	Keys and values of are corrupted in LiveCode 7
13336	mobilePixelDensity returns 100+ digit number
13335	set the textFont crashes Android
13332	Field allows line break on non-breaking space
13329	Cannot import photo into stack on Android device.
13323	'there is a file ' is not always right
13316	Setting line chunk properties on multiple lines doesn't work
13315	textDirection does not survive save & load
13314	Inconsistent Line breaks using html text
13312	Setting read-only global properties crashes Livecode

13311	Flagged block index incorrect
13300	'Set the menubar to ' causes crash
13297	Combine by Column broken in LC7DP10
13296	"the processor" returns "arm" on RaspberryPi
13294	Fix a Linux MPlayer crash
13289	Set the statusiconmenu - when used
13288	controlAtScreenLoc always returns the card
13276	abbreviated name isn't understood anymore
13263	deleted field text visible in lc7
13259	Fix system time formatting on Windows
13258	null after file name in lc7 drag drop
13255	Script debugger points to empty script when unknown XML parse error occurs
13249	tabbed data in list mode does not hilite hilitedLine correctly LCDP10
13247	Setting large htmltext is very slow
13239	iOS hard crash when using encyption
13219	Crash in OSX locale caching
13214	Hang when creating a player
13204	effective hiliteColor has changed behaviour in LC7DP9
13200	LC7 cannot save the title of stack
13186	Name comparison failure when using menuPick from tab panel
13179	Crash when getting mac resources
13042	Alt- combinations don't generate the correct character.
12903	
12776	'The number of elements of tVar' for non-array tVar hangs LC7
12547	Make arrayEncode encode in 7.0 format by default
12539	Don't draw tab characters
12502	Fix a null-pointer deref in PDF printing
11971	Password protected stacks are corrupted by LiveCode 7

Specific bug fixes (7.0.0-dp-10)

13178	Player won't play from server
13177	start using fails in livecode 7 server
13176	core image visual effects broken in LC7DP9
13146	Print to PDF fails in 7DP9
13145	ImageData display by reference hangs 7DP9
13144	answer files behaviour is broken in 7DP9
13143	LC7dp9 replaces mainStack name with /Applications in Save As dialog
13139	Incorrect parsing of
13135	Ensure that setting or getting custom properties with an index triggers the appropriate SetProp/GetProp
13124	cursor split in certain conditions in tabbed data field
13108	text selection in columnar data incorrect
13106	tabbed text with vGrid on in right align or centered mode flows over to the left
13077	Setting htmltext of field chunks can cause unexpected block order switching

Specific bug fixes (7.0.0-dp-9)

13122	Break stopped working in if statements within switch
13115	[[player]] player missing formattedwidth and formattedheight properties
13103	option
13100	LC7 DP8 Combo box label anomoly
13097	Image with no filename is not blank
13090	LC7 DP8 Split by column fails to honour blank lines
13089	Setting text of a combobox does not set the label
13084	LiveCode crashes when selecting PDF printer in printer dailog Windows desktop
13082	imageSource sometimes can't be deleted
13081	Prevent crash when evaluating non-container chunk
13079	select before | after text selects all text of field
13076	text in field does not change color when textColor property is set
13070	Fix a pointer cast that broke copy-and-paste in 64-bit builds
13057	Unable to change to initial orientation after changing orientation of device
13056	arrayDecode no longer throws an error on invalid input
13050	arrayDecode causes error when encoded array contains binary elements
13043	Stack gets corrupted after removing it from memory
13027	System icon shows rather than LiveCode icon when changing application
9058	Unmaximise windows on Linux if the max width/height is exceeded
8637	Make the "hidepalettes" property work on Linux

Specific bug fixes (7.0.0-dp-8)

13029	Windows statusiconmenu not parsed correctly
13024	Launch URL fails to launch text documents
13022	Clear Linux backdrop window after changing background colour
13018	Split by ~~and is broken with Unicode~~
12998	"Exit" is too in menu "File" on Mac
12984	setting the callback of a player crashes LiveCode
12983	Crash when looking for qteffects
12981	Clear "transient for" hint when clearing Linux backdrop
12972	Player filename dialog does not allow audio files to be selected
12952	tabbed date incorrectly displayed when vertical lines on
12951	text selection in tabbed text inconsistent
12948	Crash when opening custom property inspector having a property with more than 65535 bytes
12945	Problems with tabStops property
12937	param() is not parsed
12936	Video player crash when setting callbacks
12931	Prevent Linux backdrop from gaining focus
12925	Text - > Align does nothing
12924	Setting the style
12921	Install 32-bit and 64-bit Linux engines to different paths
12918	Object -> Flip Image on an image with a filename crashes
12916	Closing the Page Setup dialog causes a crash
12910	Script editor crashes

12909	Fix a crash on Linux when taking a snapshot of the screen
12907	File > Import as control > Snapshot from screen
12905	Set Linux geometry hints on window creation
12901	Object colors not selectable in inspector
12896	Cursor navigation broken in tabbed fields
12893	Crash when dragging away from player icon in Tools palette
12874	revBrowser (both original and CEF) crashes LiveCode 7.0 DP7
12847	Property inspector's selection menu is broken
12846	Property Inspector updates too often when moving a control
12843	thumposition returns decimal value in LC7 dp6
12729	Token chunk expression is not allowing for quotes correctly
12162	Inconsistent handling of PS in 'put into' and 'put after'

Specific bug fixes (7.0.0-dp-7)

12823	Selecting subsequent cells in a tabbed field results in incorrect highlighting
12814	Setting textDirection should force field recalculation
12797	filter with regex not working
12795	'The number of elements of tVar' for non-array tVar hangs LC7
12792	Pasting text from Text Edit into field creates gibberish
12790	Ctrl-m does not close the message box
12789	Clicking on stack listed in Application Browser causes crash
12778	Double clicking in the script editor doesn't highlight words
12777	Copy command crashes in release mode
12733	Error when getting or setting char chunk properties of buttons
12721	keyUp keyname returns gibberish
12700	Launch URL not working on LC7 in Android and iOS emulators
12697	Setting tabStop less than the preceding one on a field causes text to overlap
12695	Android video does not display
12676	Adding number to numeric value in variable gives incorrect result on LC7
12672	LC 7.0DP6 Crash on Save After Editing Large Script
12659	Error on Android when reading files list from the stack folder path
12656	Decomposing native strings doesn't work
12651	back key can not work
12650	Copying externals files to android app fails
12644	Filtering unicode text with wildcard can result in false positives
12610	Split by column causes crash
12596	Number of controls of card returns wrong value if given a card id
12595	Printing to PDF does not yield all information
12576	drawing_bug_when_rotating_graphic
12574	REGEX : matchText result not as expected
12562	Changing the back color of a line which contains a tab makes LC crash
12552	go to url internet stack path does not work
12540	Clipboarddata should return utf16 data for 'unicode' mode
12538	Read from process until empty
12532	Adding a new element to an array can be very slow

12488	Tabbed characters are cut off on the left
12478	Retrieving data from url results in garbled data on iOS from LiveCode 7
12343	Hebrew text is shown in reverse character order on Android
12166	Fix cursor movement over zero-width characters

Specific bug fixes (7.0.0-dp-6)

12544	send command with a parameter which contains a quote breaks param parsing
12530	embedded wav sound crashes Project Browser and Properties inspector in LC 7 dp5
12527	paragraph chunk returns empty when string does not include end of paragraph mark
12521	Fix highlights for non-left-aligned lines in fields
12517	Quicktime using stacks crash on open
12515	crash on clicking linktext (on second click)
12514	dragData with a private content extracted from a string by using a chunk keyword (word
12511	charIndex property missing
12510	setting stack decoration errors
12509	fullscreenMode "showAll" breaks IDE
12493	open file for binary read/write erroneously converting line endings
12477	Native mobile controls created with mobGui do not seem to function under LiveCode 7.0

Specific bug fixes (7.0.0-dp-5)

12499	trueWord n + m of tText for n the number of trueWords of tText always returns trueWord n
12497	pageRanges property missing from LiveCode 7.0
12496	[[Bugfix 12496]] Set the clipping rectangle for text blocks correctly
12494	Setting the randomSeed to large number fails in 7.0
12491	"Go to Definition" doesn't work in script editor
12489	filter/replace difference in 7.0
12486	[[Bugfix 12486]] Add missing MovieControllerID property to the Player property table
12483	Graphic effects not working in 7.0 DP4
12482	replace does not work
12074	Answer dialog messages should be aligned to the right

Specific bug fixes (7.0.0-dp-4)

12459	Setting any graphic effects to "none" crashes LC 7 dp3
12457	sorting marked cards with single unmarked card crashes LiveCode
12432	clickchunk and click text are not identical
12428	Lc 7.0 DP3 does not sanitize data when setting points of polygon
12423	If you choose the browse tool (run) after Editing a group - Livecode crashes.
12422	Sort puts a "p" after the last character and foreign letters is not sorted correct
12409	Fields in LC 7 fail to display binfile url imagesource
12407	'Garbage' with read from socket
12360	open file as utf-8 mode doesn't work exactly as documented
12345	AVD's appear in the list but can't be selected for testing.
12344	Can't open recent file
12309	Build for Windows fails with i/o error
12288	Prevent User Samples stack hanging due to resize error

12246	Serial I/O fails on write
12192	linux uninstaller needs execute permission
12061	Can't test an app on Android
11989	arrayDecode on a file containing the result of arrayEncode on an empty array causes execution error

Specific bug fixes (7.0.0-dp-3)

12290	saving 2.7 file format stack causes crash
12244	case sensitive does not work
12204	textEncode ASCII support is actually native
12195	equality testing is slow
12194	'char/byte/codepoint 1 of s' is slow
12184	'repeat for each byte b in empty' crashes
12180	'the number of bytes of ...' is slow
12179	Fetching byte chunks does not clamp the range to the bounds of the input data.
12168	Sometimes length() and number or chars are wrong
12160	Put after/before on an uninitialised
12150	LiveCode crashes when changing the window kind
12147	create button in group command fails
12143	The mousechunk end index is one larger than it ought to be
12140	Erroneous Socket Timeout Error
12138	the drawer command crashes Livecode 7.0 when using '...at position' variant.
12123	Fix wrong application title displaying on Linux
12122	Update GTK icon cache post-install
12118	revExecuteSQL writes incomplete data into SQLite BLOB columns
12078	Scrambled word order for label field with Hebrew and English Text
12075	Buttons that contain Hebrew Text is in wrong order
12007	Linux Standalone does not run. Segmentation fault.
11993	"save stack" corrupt password protected stacks
11979	IDE fails to launch when installed to a Unicode path
11973	char 1 of (e + combining acute accent) returns e
11962	Split command causes IDE to stop responding
11961	IDE takes 8 seconds when adding a new line in Script Editor
11941	repeat loop is very slow in 7.0 DP1
11939	Opening the TestFramework stack crashes LiveCode

Specific bug fixes (7.0.0-dp-2)

12104	Convert command fails with invalid date since 7.0
12097	setting acceleratorModifiers of button causes crash
12081	OSX picking wrong file extension for filenames with two '.' characters
12071	hiliteColor and borderColor is not working in 7.0DP1
12070	hGrid
12067	Group with label can't be saved in 5.5 file format
12065	formatting hex string crashes LiveCode 7.0
12042	New chunk types (paragraph

12038	'lock screen for visual effect in rect...' not working
11996	numToByte works differently form numToChar in 6.6
11985	put does not populate the result on iOS
11981	calling mobileControlTarget () crashes the application
11963	Dotted border of selection in List control is incorrectly aligned
11960	LC crashes when selecting wrapped text in Contents pane
11958	Text wrapping improperly breaks text mid-word
11954	sort field does not work
11953	sort card of stack crashes
11950	mark card does not work
11949	find string in field does not work
11948	Export snaphot crashes LiveCode when it should return empty rect error
11947	Vertical tabulation in a field causes the engine to hang
11945	The number of paragraphs reported value is incorrect
11943	Script Editor does not resize correctly with the resize handle
11940	Variables not being resolved in the script debugger.

Dictionary additions

- **byteOffset** (*function*) has been added to the dictionary.
- **codepointOffset** (*function*) has been added to the dictionary.
- **codepointProperty** (*function*) has been added to the dictionary.
- **codepointToNum** (*function*) has been added to the dictionary.
- **codeunitOffset** (*function*) has been added to the dictionary.
- **nativeCharToNum** (*function*) has been added to the dictionary.
- **normalizeText** (*function*) has been added to the dictionary.
- **numToCodepoint** (*function*) has been added to the dictionary.
- **numToNativeChar** (*function*) has been added to the dictionary.
- **paragraphOffset** (*function*) has been added to the dictionary.
- **sentenceOffset** (*function*) has been added to the dictionary.
- **textDecode** (*function*) has been added to the dictionary.
- **textEncode** (*function*) has been added to the dictionary.
- **tokenOffset** (*function*) has been added to the dictionary.
- **truewordOffset** (*function*) has been added to the dictionary.
- **codepoint** (*keyword*) has been added to the dictionary.
- **codepoints** (*keyword*) has been added to the dictionary.
- **codeunit** (*keyword*) has been added to the dictionary.
- **codeunits** (*keyword*) has been added to the dictionary.
- **paragraph** (*keyword*) has been added to the dictionary.
- **paragraph** (*keyword*) has been added to the dictionary.
- **segment** (*keyword*) has been added to the dictionary.
- **segments** (*keyword*) has been added to the dictionary.
- **sentence** (*keyword*) has been added to the dictionary.
- **sentences** (*keyword*) has been added to the dictionary.
- **trueWord** (*keyword*) has been added to the dictionary.
- **trueWords** (*keyword*) has been added to the dictionary.
- **cursorMovement** (*property*) has been added to the dictionary.
- **formSensitive** (*property*) has been added to the dictionary.

- **tabAlign** (*property*) has been added to the dictionary.
- **textDirection** (*property*) has been added to the dictionary.

Dictionary changes

- The entry for **mobilePickPhoto** (*command*) has been updated.
- The entry for **open driver** (*command*) has been updated.
- The entry for **open file** (*command*) has been updated.
- The entry for **open process** (*command*) has been updated.
- The entry for **revBrowserSet** (*command*) has been updated.
- The entry for **sort container** (*command*) has been updated.
- The entry for **sort** (*command*) has been updated.
- The entry for **repeat** (*control structure*) has been updated.
- The entry for **arrayEncode** (*function*) has been updated.
- The entry for **charToNum** (*function*) has been updated.
- The entry for **longFilePath** (*function*) has been updated.
- The entry for **measureUnicodeText** (*function*) has been updated.
- The entry for **mobileLocationAuthorizationStatus** (*function*) has been updated.
- The entry for **numToChar** (*function*) has been updated.
- The entry for **revBrowserOpenCef** (*function*) has been updated.
- The entry for **uniDecode** (*function*) has been updated.
- The entry for **uniEncode** (*function*) has been updated.
- The entry for **byte** (*keyword*) has been updated.
- The entry for **character** (*keyword*) has been updated.
- The entry for **word** (*keyword*) has been updated.
- The entry for **words** (*keyword*) has been updated.
- The entry for **is among** (*operator*) has been updated.
- The entry for **is not among** (*operator*) has been updated.
- The entry for **stackFileVersion** (*property*) has been updated.
- The entry for **umask** (*property*) has been updated.
- The entry for **unicodeFormattedText** (*property*) has been updated.
- The entry for **unicodeLabel** (*property*) has been updated.
- The entry for **unicodePlainText** (*property*) has been updated.
- The entry for **unicodeText** (*property*) has been updated.
- The entry for **unicodeTitle** (*property*) has been updated.
- The entry for **unicodeTooltip** (*property*) has been updated.
- The entry for **useUnicode** (*property*) has been updated.

Previous Release Notes

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
6.0.2 Release Notes	http://downloads.livecode.com/livecode/6_0_2/LiveCodeNotes-6_0_2.pdf
6.0.1 Release Notes	http://downloads.livecode.com/livecode/6_0_1/LiveCodeNotes-6_0_1.pdf
6.0.0 Release Notes	http://downloads.livecode.com/livecode/6_0_0/LiveCodeNotes-6_0_0.pdf