

LiveCode 7.0.0-dp-8 Release Notes

Table of contents

- Overview
- 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
- Proposed changes
- Engine changes
 - Effective points of graphics
 - Player messages aren't sent correctly.
 - Don't draw tab characters
 - Hebrew text is shown in reverse character order on Android
 - Password protected stacks are corrupted by LiveCode 7
 - Unicode Support
 - Unicode and LiveCode
 - Creating Unicode Apps
 - New & Existing apps - things to look out for
- New Commands, Functions & Syntax
 - Chunk expressions: byte, char, codepoint, codeunit
 - Chunk expressions: paragraph, sentence and trueWord
 - Synonym: segment
 - Property: the formSensitive
 - Command: open file/process/socket ... for <encoding> text
 - Functions: textEncode, textDecode
 - Functions: numToCodepoint, codepointToNum
 - Functions: numToNativeChar, nativeCharToNum
 - Function: normalizeText
 - Function: codepointProperty
- Updated Functions
- Function: binaryEncode
- Function: binaryDecode
- Deprecated Features
 - Functions: numToChar, charToNum
 - Property: useUnicode
 - Functions: uniEncode, uniDecode
 - Function: measureUnicodeText

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

Specific bug fixes (7.0.0-dp-8)

Specific bug fixes (7.0.0-dp-7)

Specific bug fixes (7.0.0-dp-6)

Specific bug fixes (7.0.0-dp-5)

Specific bug fixes (7.0.0-dp-4)

Specific bug fixes (7.0.0-dp-3)

Specific bug fixes (7.0.0-dp-2)

Dictionary additions

Dictionary changes

Previous Release Notes

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 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:

- 32-bit installation, or a 64-bit linux distribution that has a 32-bit compatibility layer
- 2.4.x or later kernel
- X11R5 capable Xserver running locally on a 24-bit display
- glibc 2.3.2 or later
- gtk/gdk/glib (optional – required for native theme support)
- pango/xft
- lcms (optional – required for color profile support in JPEGs and PNGs)
- gksu (optional – required for elevate process support)

Note: The optional requirements (except for gksu and lcms) are also required by Firefox and Chrome, so if your linux distribution runs 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: LiveCode and standalones it builds may work on remote Xservers and in other bit-depths, however this mode of operation is not currently 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-dp-8
- Linux: /opt/runrev/livecode-7.0.0-dp-8
- Mac: /Applications/ LiveCode 7.0.0-dp-8.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-dp-8
- Linux: ~/.runrev/components/livecode-7.0.0-dp-8
- Mac: ~/Applications/ LiveCode 7.0.0-dp-8.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-dp-8
```

- 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>/<user>/<user>/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 <i>location</i>	The location to install into. If not specified, the location defaults to those described in the <i>Layout</i> section above.
-log <i>logfile</i>	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

Effective points of graphics (7.0.0-dp-8)

You can now use 'the effective points' and 'the effective relativePoints' properties of a graphic object to fetch a polygon representation of rectangle, round rectangle, oval and regular polygon graphic objects.

Player messages aren't sent correctly. (7.0.0-dp-8)

The occurrence of playStarted, playPaused and playStopped messages has been cleaned up.

The playStarted message will only be sent when the rate of the movie changes from zero to non-zero - whether via clicking the play button, setting the playRate or by using play start / play resume via script.

The playPaused message will only be sent when the rate of the movie changes from non-zero to zero - whether via clicking the pause button, setting the playRate or by using play pause / play stop.

The playStopped message will only be sent when the movie reaches the end of playback.

These are the only cases in which the messages will be sent - in particular, setting the filename will no longer send any messages and you will not get multiple messages of the same type in succession.

Don't draw tab characters (7.0.0-dp-7)

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.

[[Bugfix 12502]] Fix a null-pointer deref in PDF printin (7.0.0-dp-5)g

Password protected stacks are corrupted by LiveCode 7 (7.0.0-dp-2)

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 Supplementary 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-dp-8)

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

- 13029 Windows statusiconmenu not parsed correctly**
- 13025 Linux GTK spinboxes were inverted**
- 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~~**
- 13011 currentTimeChanged not sent when dragging controller thumb.**
- 13010 If an object is in a group then mouseStillDown will not fire if it is in a behavior of the object.**
- 13009 ImageData copying to offscreen images is broken**
- 13007 Native encoded characters > 127 don't appear in menuPick string.**
- 13005 Modifying selection by shift + click in the controller doesn't work correctly.**
- 13004 Controller not updated when setting currentTime by script.**
- 13003 selectionChanged message not sent when player selection changes via controller.**
- 13002 Empty selection results in selection of full movie.**
- 13001 Crash when changing focus inside openField handler.**
- 13000 Image colours not showing correctly on Android device**
- 12998 "Exit" is too in menu "File" on Mac**
- 12997 Changing windowShape leaves artifacts behind.**
- 12990 Popup menus always highlight first item on Mac**
- 12989 Player incorrectly reports timeScale and duration when using AVFoundation.**
- 12984 setting the callback of a player crashes LiveCode**
- 12983 Crash when looking for qteffects**
- 12981 Clear "transient for" hint when clearing Linux backdrop**
- 12979 Setting the rect of a player object doesn't work correctly if the stack is pixelScaled or scaled in any fullscreen mode.**
- 12978 Development -> Plugins opens the wrong plugin**
- 12972 Player filename dialog does not allow audio files to be selected**

- 12968 ctrl z and edit -> undo are not working
- 12963 Player messages aren't sent correctly.
- 12962 Player won't play video in certain circumstances.
- 12961 Print dialogs do not keep new settings if displayed as sheet.
- 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
- 12946 play stop command does not work.
- 12945 Problems with tabStops property
- 12944 Text does not align at correct tabStops in some cases.
- 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
- 12923 Setting the text style in the Text menu does not update correctly
- 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
- 12867 Gradient colours display incorrectly on android
- 12865 LiveCode Server process does not end when run as CGI
- The contents of a folder added to the Copy Files section of the Standalone Application
- 12864 Settings are not copied during a standalone build when it's a subfolder of the defaultFolder and the defaultFolder contains the main stack file.
- 12847 Property inspector's selection menu is broken
- 12846 Property Inspector updates too often when moving a control
- 12844 Attempted to arrayDecode non-arrayEncoded data can cause a crash.
- 12843 thumposition returns decimal value in LC7 dp6
- 12822 LiveCode server fails to read from https URLs on Fedora
- 12819 revAvailableHandlers works on password protected stacks.
- 12804 Clipping incorrect when printing to PDF.
- 12729 Token chunk expression is not allowing for quotes correctly
- 12594 NUL bytes in fields will cause printing to pdf to fail silently.
- 12429 Setting the defaultNetworkInterface more than once causes instability.
- 12298 Autocomplete text in Message Box does not line up with text you are typing
- 12250 File view in file dialog doesn't update if you change the filtered types.
- 12162 Inconsistent handling of PS in 'put into' and 'put after'

- 11828 Use 64-bit file APIs on Linux**
- 11797 scriptLimits no longer enforced**
- 9405 Clear previous highlight before drawing Linux GTK button highlight**

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
- 12539 Don't draw tab characters
- 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)

- 12502 [[Bugfix 12502]] Fix a null-pointer deref in PDF printing
- 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 of 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
- 11971 Password protected stacks are corrupted by LiveCode 7
- 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 snapshot 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 **do** (*command*) has been updated.

- The entry for **edit** (*command*) has been updated.
- The entry for **insert script** (*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 **backScripts** (*function*) has been updated.
- The entry for **charToNum** (*function*) has been updated.
- The entry for **frontScripts** (*function*) has been updated.
- The entry for **longFilePath** (*function*) has been updated.
- The entry for **measureUnicodeText** (*function*) has been updated.
- The entry for **numToChar** (*function*) has been updated.
- The entry for **revBrowserOpenCef** (*function*) has been updated.
- The entry for **scriptLimits** (*function*) has been updated.
- The entry for **uniDecode** (*function*) has been updated.
- The entry for **uniEncode** (*function*) has been updated.
- The entry for **** (*keyword*) has been updated.
- The entry for **byte** (*keyword*) has been updated.
- The entry for **character** (*keyword*) has been updated.
- The entry for **;** (*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 **points** (*property*) has been updated.
- The entry for **script** (*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