LiveCode 4.6.1 Release Notes

Table of Contents

Overview

LiveCode 4.6.1 is an update to 4.6 containing bug fixes, minor enhancements and support for deploying Android applications.

This document describes all the changes that have been made – including bug fixes and new syntax.

For information about improvements made to the iOS port of the engine, please see the iOS Release Notes PDF.

At the time of writing, this information has yet to been integrated into the dictionary or User's Guide.

Known issues

- The installer will currently fail if you run it from a network share on Windows. Please copy the installer to a local disk before launching on this platform.

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 2000 SP4
- 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

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 (optional – required for pdf printing, anti-aliased text and unicode font support)

- 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:

- 10.4.11 (Tiger) on Intel and PowerPC

- 10.5.8 and later (Leopard) on Intel and PowerPC

- 10.6.x (Snow Leopard) on Intel

- 10.7.x (Lion) on Intel (experimental)

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

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, glib, pango and xft on Linux

Setup

Installation

The structure of the IDE install has changed significantly in this release.

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 4.6.1

* Linux: /opt/runrev/livecode-4.6.1

* Mac: /Applications/ LiveCode 4.6.1.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 4.6.1

* Linux: ~/.runrev/components/livecode-4.6.1

* Mac: ~/Applications/ LiveCode 4.6.1.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-4.6.1
  ```

* 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 license system has been replaced in this release.

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

On both platforms, the command is of the form:

> <exe> activate -file *license* -passphrase *phrase*

This command will load the manual activation file from *license*, decrypt it using the given *passphrase* and then install a license file for all users of the computer. Manual activation files can be downloaded from the 'My Products' section of the RunRev customer accounts area.

This action can be undone using the following command:

> <exe> deactivate

Again, as the LiveCode executable is actually a GUI application it needs to be run slightly differently from other command-line programs.

In what follows <livecodeexe> should be replaced with the path to the installed LiveCode executable or app that has been previously installed.

On Windows, you need to do:

> start /wait <livecodeexe> activate -file *license* -passphrase *phrase*

> start /wait <livecodeexe> deactivate

On Mac, you need to do:

> "<livecodeexe>/Contents/MacOS/LiveCode" activate -file *license* -passphrase *phrase*

> "<livecodeexe>/Contents/MacOS/LiveCode" deactivate

On both platforms, the result of the activation will be written to the console.

Engine Changes

Print to PDF document outline support (4.6 – improved 4.6.1)

You can now create a document outline in PDFs generated by the pdf printing commands. Individual entries in the document outline are referred to as 'bookmarks' and can be created with the following syntax:

print [**unicode**] **bookmark** *title* [**with level** *level*] [**at** *location*] [**initially** (**open** | **closed**)]

Here *title* is the name of the bookmark which will be displayed, *level* is the depth at which the entry should be placed, and *location* is a point describing where on the current page the bookmark should refer to.

If the unicode adjective is specified, then *title* will be interpreted as a UTF-16 encoded string, otherwise it is interpreted as the native (single-byte) text encoding.

If *level* is not present, then the entry is inserted at the level of the previously inserted item, defaulting to the top-most level if no previous entries have been made.

If *location* is not present, the top-left of the current page will be taken as the location.

If an initially clause is present, it determines the default 'open' state of the bookmark item. Bookmarks are, by default, open. *(4.6.1)*

Mac – revCapture video capture library (4.6.1 – experimental)

The revCapture library is begin developed as a replacement for revVideograbber – utilising the most up to date APIs on each platform, and eliminating many issues the previous library had (including problems introduced by Apple with iSight video-capture in 10.6.5 onwards).

At this time, the revCapture library is Mac-only, and is integrated as part of the revVideograbber external; it will be moved into its own external and become cross-platform over time.

While largely similar in function to revVideograbber, the revCapture library does have a number of differences:

- There are no configuration dialogs – instead, available inputs and codecs can be queried and configured through script.

- Previewing and recording can be started and stopped independently.

- Previewing takes place direct into a (configurable) image object, allowing the preview to be rendered with effects, blending, inks and interleave with other objects.

- Only supports output to QuickTime MOV files with a limited collection of preconfigured codecs.

The available handlers are described in the following sections.

Note: As it stands the revCapture library has only been tested with built-in iSight hardware, although there is no reason it shouldn't work with any hardware that QTKit supports.

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Session management

All capture operations take place within a capture session, which is managed using:

>**revCaptureBeginSession**

>**revCaptureEndSession**

Call **revCaptureBeginSession** to initialize the capture session, allowing the rest of the capture library to be used. When the capture session is done with, call **revCaptureEndSession**. This will release any hardware, and cancel any recording and previewing operations.

*Note: Calling any of the revCapture functions without calling **revCaptureBeginSession** at some point before hand will cause a no session error to be thrown.*

Input configuration

Unlike revVideograbber, configuration of the inputs to use for video/audio capture is done entirely through script rather than invoking standard dialogs.

Querying available inputs

To query the currently available inputs capable of capturing audio or video data use:

>**revCaptureListAudioInputs()**

>**revCaptureListVideoInputs()**

These both return a newline-delimited list of the available inputs of the given type.

Configuring the audio input

To configure the audio input to capture from in the current session, use:

>**revCaptureGetAudioInput()**

>**revCaptureSetAudioInput** *audioInput*

Where *audioInput* is one of:

- the name of an input returned by **revCaptureListAudioInputs()**

- *default*, indicating the default audio input device on the system should be used

- *none*, indicating that no audio should be captured

If setting an audio input fails because the new device could not be accessed, a *could not open input* error is thrown. If the current session could not connect to the new device, a *could not connect to input* error is thrown.

Configuring the video input

To configure the video input to capture in the current session, use:

>**revCaptureGetVideoInput()**

>**revCaptureSetVideoInput** *videoInput*

Where *videoInput* is one of:

- the name of an input returned by **revCaptureListVideoInputs()**

- *default*, indicating the default video input device on the system should be used

- *none*, indicating that no video should be captured

If setting an audio input fails because the new device could not be accessed, a *could not open input* error is thrown. If the current session could not connect to the new device, a *could not connect to input* error is thrown.

Compression configuration

Similar to input configuration, choosing the codec to use when recording audio and video streams is done entirely through script.

Querying available codecs

To query the currently available codecs for audio or video recording use:

revCaptureListAudioCodecs()

revCaptureListVideoCodecs()

These both return a newline-delimited list of the available codecs of the given type.

Configuring the audio codec

To configure the audio codec to use when recording, use:

revCaptureGetAudioCodec()

revCaptureSetAudioCodec *audioCodec*

Where *audioCodec* is either *none* to indicate that no audio compression should be performed, or the name of a codec returned by **revCaptureListAudioCodecs()**.

If the given codec name is unrecognized, or is not suitable for audio compression then an *invalid codec for media type* error is thrown. If recording is currently happening, a *recording in progress* error is thrown.

Configuring the video codec

To configure the video codec to use when recording, use:

revCaptureGetVideoCodec()

revCaptureSetVideoCodec video*Codec*

Where *videoCodec* is either *none* to indicate that no video compression should be performed, or the name of a codec returned by **revCaptureListVideoCodecs()**.

If the given codec name is unrecognized, or is not suitable for video compression then an *invalid codec for media type* error is thrown. If recording is currently happening, a *recording in progress* error is thrown.

Preview management

Configuring the video preview image

In order to be able to see a video preview of a capture session, the long id of an image object to target needs to be provided. To manage this use:

revCaptureGetPreviewImage()

revCaptureSetPreviewImage *imageLongId*

Where *imageLongId* is the long id of an existing empty image object to use; or empty to turn off the preview image.

Setting the preview image will lock the corresponding object meaning it can't be resized or have it's content modified by anything apart from the capture session. If the image cannot be found, locked, or an offscreen buffer initialized a suitable error will be thrown.

Configuring the audio preview volume

To configure the volume of the audio preview of a capture session use:

revCaptureGetPreviewVolume()

revCaptureSetPreviewVolume *newVolume*

Where *newVolume* is an integer between 0 (no audio preview) and 100 (audio preview at maximum volume).

Previewing a session

To start or stop a preview from running use:

revCaptureStartPreviewing

revCaptureStopPreviewing

The status of the preview is independent to that of recording, meaning that you can preview without recording, and record without previewing.

If a preview of the session could not be initialized when calling **revCaptureStartPreviewing** a *could not connect to output* error is thrown.

Note: Starting to record may have an effect on the video preview's aspect and frame size as the video capture pipeline is optimized by the system based on the needs of the recorded output.

Record management

Configuring the output file

To configure the filename to use when recording a capture session use:

revCaptureGetRecordOutput()

revCaptureSetRecordOutput *recordFilename*

Where *recordFilename* is an (absolute) path to the file to use for recording.

If recording is currently in progress, a *recording in progress* error is thrown.

Configuring the maximum frame size

To configure the maximum size of a frame in the recorded file use:

revCaptureGetRecordFrameSize()

revCaptureSetRecordFrameSize *maxWidth, maxHeight*

Where *maxWidth* and *maxHeight* specify the maximum size of a frame (in pixels) that should be generated in the (compressed) output. To use the default (optimal) frame size for the current settings, specify both width and height as 0.

Note: These act as a hint to the pipeline, and the actual size of the frames in the output file may be smaller than this, or have a different aspect ratio.

Configuring the maximum frame rate

To configure the maximum frame rate to aim for in the recorded file use:

revCaptureGetRecordFrameRate()

revCaptureSetRecordFrameRate *maxFrameRate*

Where *maxFrameRate* is the number of frames per second to aim for. To use the default (optimal) frame rate for the current settings, specify the rate as 0.

Note: This acts as a hint to the pipeline as to the rate to aim for. The resulting frame rate in the recorded file may be less than, or more than this setting.

Starting recording

To start recording the current session use:

revCaptureStartRecording

This configures the session with previously selected record options and codecs, deletes the currently specified output file (if present) and starts the recording process. It has no effect if the session is already being recorded.

If recording starts successfully, **the result** is empty.

If an error occurs when the output file is being prepared for recording, **the result** will contain *recording failed*.

If the session could not be configured for recording, a *could not connect to output* error is thrown.

Stopping recording

To stop recording the current session use:

revCaptureStopRecording

This finishes outputting any pending samples to the output file and stops recording. It has no effect if the session is not currently being recorded.

If recording finished successfully, **the result** is empty.

If an error occurs while trying to finish writing the output file, **the result** will contain *recording failed.*

Object snapshot with effects (4.6)

The import and export snapshot commands have been augmented with the ability to specify whether graphic effects and the blendLevel should be applied to the object before rending.

The revised syntax of these commands is:

> **import snapshot from** [**rectangle** *rectangle* **of**] *object* [(**with** | **without**) **effects**]

> **export snapshot from** [**rectangle** *rectangle* **of**] *object* [(**with** | **without**) **effects**] …

Where … denotes the usual possible suffixes for the export snapshot command.

If the effects clause is not specified, then the object is rendered without effects.

When rendering with effects *rectangle* is clipped to the maximum rectangle the object needs to fully render all the applied effects (this may be considerably bigger than the object, is a distant drop-shadow is involved).

Print to PDF link improvements (4.6)

The syntax for the 'print link' command has been changed to:

> **print link to** [**url** | **anchor**] *link* **with rect** *rectangle*

If the url adjective is specified, then *link* will be interpreted as a URI and will create an external URI launching link in the PDF document.

If the anchor adjective is specified, then *link* will be interpreted as an internal anchor name and will create an internal anchor link in the PDF document.

If no adjective is specified then *link* will be interpreted as a URI if it begins with (ASCII) letters followed by a colon. e.g. "mailto:foobar" would be interpreted as a URI, whereas "mail-to-anchor:foo" would be interpreted as an anchor name.

Field 'plain text' properties (4.6)

Several field properties related to plain text export have been added or updated in the field object.

The *formattedText* and *unicodeFormattedText* properties return the contents of the field as plain text after any wrapping has been performed, where soft (or explicit) line-breaks are exported as a single return character, and paragraph-breaks are exported as two return characters. Any paragraphs with listStyle set get prefixed by an appropriate plain-text form of the bullet or index.

The *plainText* and *unicodePlainText* properties return the contents of the field as plain text with any listStyle properties being converted appropriately into plain-text for the paragraphs they affect.

In both cases the unicode property variants return text encoded in UTF-16 in host byte-order, where as the non-unicode properties return the text encoded in the native platform encoding (with '?' replacing any non-convertible characters).

15

Out-of-bounds group scrolling (4.6)

It is now possible to configure a group to turn off automatic clamping of scroll offsets when scrollbars are not present. To control this use the following properties:

> **set the unboundedHScroll of** *group* **to** *booleanValue*

> **set the unboundedVScroll of** *group* **to** *booleanValue*

When an unbounded scroll property is set to true and the corresponding scrollbar is not visible, it is possible to set the scroll for that axis to values outside that of the size of the content (both positive and negative).

The main motivation for this change is to allow much easier implementation of the 'bouncing' effect common in iOS controls. Indeed, after setting the relevant unbounded scroll properties, it is enough to just use the scroll value provided by a scroller object directly as the value passed to the relevant group scroll property.

If a scrollbar is made visible or unbounded scroll is turned off and the corresponding scroll property is outside of the content bounds, the group is scrolled to bring the value within the appropriate range.

Cryptographic-quality random data (4.6)

A **randomBytes()** function has been added to enable generation of arbitrary amounts of cryptographic-quality random data. To use this feature, the syntax is:

> **randomBytes(***byteCount***)**

The function will return *byteCount* random bytes and uses the OpenSSL library's random data generator to do so.

The random data is derived from non-predictable sources where possible, meaning that it can (for the most part) be considered truly random. This is contrast to the **random** function which uses a pseudo-random number generator and **randomSeed** value.

Note: Make sure the security library is included when building applications that use this function as a standalone. If the library cannot be found at runtime, the function with throw an error.

SHA-1 digest support (4.6)

The SHA-1 digest of a block of data can now be computed by using the following form:

> **the sha1Digest of** *data*

> **sha1Digest(***data***)**

This function returns the sha-1 digest in the form of 20 (binary) bytes.

Public key encryption support (4.6)

The **encrypt** and **decrypt** commands now support RSA public key encryption. The new forms supporting this are:

> **encrypt** *message* **using rsa with (public | private) key** *key* **[and passphrase** *passphrase* **]**

> **decrypt** *message* **using rsa with (public | private) key** *key* **[and passphrase** *passphrase* **]**

Here *key* should be in PEM format, optionally protected by *passphrase*.

The maximum length of a message that can be encrypted using RSA is the size of the key in bytes – 11. So, for a 512-bit key pair, the maximum encryptable message size is 53 bytes.

Generating key pairs

Public-private key pairs can be generated using the **OpenSSL** suite of command-line tools. For example:

> **openssl genrsa -out** *private_key.pem* 512

> **openssl rsa -pubout -in** *private_key.pem* **-out** *public_key.pem*

Will generate a key pair of size 512-bits, placing the private key in *private_key.pem* and the public key in *public_key.pem*.

For more information on these utilities see http://www.openssl.org/docs/apps/rsa.html and http://www.openssl.org/docs/apps/genrsa.html.

Command variants

Use encrypt with a public key to encode a message that you only want to be decoded the holder of the private key.

Use decrypt with a private key to decode a message that a sender has encrypted with its corresponding public key.

Use encrypt with a private key to encode a message that a receiver can then verify has come from one of the holders of the private key (this is a *signing* operation).

Use decrypt with a public key to verify that a message has been encoded with the corresponding private key, and there has come from one of its holders (this is a *verify* operation).

Encrypting longer messages

For signing, the maximum length of an encryptable message isn't really an issue since typically in that scenario it will be some sort of hash that would be being encrypted.

For the more traditional encrypting scenario, however, the standard approach is to use public key cryptography to encrypt a random password which is then used with a symmetric cipher to actually encrypt the payload.

Mac – window modification indicator (4.6)

Standard Mac document windows typically display the modified state of the document they represent by adding a dark dot to the close-box. It is possible to control the display of this dot in LiveCode by using:

> **set the modifiedMark of** *stack* **to** *booleanValue*

The indicator will be displayed when the property is set to *true*.

This property has no effect on Windows nor Linux.

XML text node access (4.6 – experimental)

Support has been added to some revXML functions for manipulating text nodes. Consider the following XML fragment:

<summary>

Removes a <keyword>message</keyword> that was queued with the <command>send</command> command and is waiting to be sent.

<summary>

This has the following structure as an XML tree:

ELEMENT(summary)

TEXT(Removes a)

ELEMENT(keyword)

TEXT(message)

TEXT(that was queued with the)

ELEMENT(command)

TEXT(send)

TEXT(command and is waiting to be sent.)

Notice that the named XML nodes, are interspersed with (essentially unnamed) nodes containing text content. These (previously unaccessible) nodes can now be accessed (with some functions) by using a path of the form:

<parent>/[<index>]

Here this references the <n>th text node under <parent>. For example, the above tree has the following accessible nodes:

summary/[1]

summary/keyword

summary/keyword/[1]

summary/[2]

summary/command

summary/command/[1]

summary[3]

To access text content of a node simply use the *revXMLNodeContents* function with this extended path format. Note that the previous behavior is preserved – if you specify a node with a '/[n]' suffix, the text content of the node will be returned (if it is a text node). (i.e summary/keyword and summary/keyword/[1] are the same thing to *revXMLNodeContents*).

The following functions have been augmented with an additional (optional) parameter *incText*:

revXMLFirstChild(docId, [incText])

revXMLNextSibling(docId, nodePath, [incText])

revXMLPrevSibling(docId, nodePath, [incText])

revXMLChildNames(docId, nodePath, delimiter, filter, incCounts, [incText])

Here if *incText* is specified and is *true*, the functions will include text nodes in their processing.

For example, this allows you to loop over all nodes *including text nodes* using something like:

local tCurrentNode

put revXMLFirstChild(tDocId, tParentNode, **true**) **into** tCurrentNode

repeat while tCurrentNode **is not empty**

    ... use tCurrentNode ...

    put revXMLNextSibling(tDocId, tCurrentNode, **true**) **into** tCurrentNode

end repeat

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Ordered and unordered lists (4.6 – experimental)

Properties

Experimental support has been added to the field for simple, single level, ordered and unordered lists. To make a paragraph display as an element in a list use the *listStyle* property:

set the listStyle of line *lineIndex* **of** *field* **to** *style*

Where *style* is one of:

- *disc* – the paragraph is rendered as an element in an unordered list with the standard bullet character as marker (U+2022).

- *circle* – the paragraph is rendered as an element in an unordered list using the (unicode) character U+25E6 as marker.

- *square* – the paragraph is rendered as an element in an unordered list using the (unicode) character U+25AA as marker.

- *decimal* – the paragraph is rendered as an element in an ordered list, the label using standard (Arabic) decimal numerals. i.e. 1, 2, 3, 4, etc.

- *lower latin* – the paragraph is rendered as an element in an ordered list, the label using lowercase (Latin) letters. i.e. a, b, c, …, aa, ab, etc.

- *upper latin* – the paragraph is rendered as an element in an ordered list, the label using uppercase (Latin) letters. i.e. A, B, C, …, AA, AB, etc.

- *lower roman* – the paragraph is rendered as an element in an ordered list, the label using lowercase Roman numerals. i.e. i, ii, iii, iv, …, etc.

- *upper roman* – the paragraph is rendered as an element in an ordered list, the label using lowercase Roman numerals. i.e. I, II, III, IV, …, etc.

For both ordered and unordered list the marker is placed at the first tab-stop, and the content of the paragraph is placed (and wraps) at the second tab-stop.

For ordered lists the index of the item is determined by the number of preceding paragraphs with the **same** *listStyle* property.

Setting the *listStyle* property of a paragraph to empty causes it to revert to a normal (non-list item) paragraph.

htmlText format

Paragraphs with a non-empty *listStyle* present themselves in *htmlText* wrapped with *<LI>* tags. Sequences of such paragraphs with the same *listStyle* are bracketed by <UL> or <OL> tags. These tags take a *type* attribute, matching the (legacy) HTML attribute of the same name:

- *disc → disc*
- *circle → circle*
- *square → square*
- *1 → decimal*
- *a → lower latin*
- *A → upper latin*
- *i → lower roman*
- *I → upper roman*

For example:

1. Numbered Item 1
2. Numbered Item 2
- Bulleted Item 1
- Bulleted Item 2

```
<ol type='1' >
<li><p>Numbered Item 1</p></li>
<li><p>Numbered Item 2</p></li>
</ol>
<ul type='disc'>
<li><p>Bulleted Item 1</p></li>
<li><p>Bulleted Item 2</p></li>
</ul>
```

rtfText format

Paragraphs with a non-empty *listStyle* are appropriately marked in *rtfText* using both the (legacy) *pn* family of paragraph numbering tags and also with the new *listtable* tags.

By using both sets of tags a reasonable degree of interoperability is achieved with both TextEdit (and other Cocoa applications) on Mac, and Word and WordPad on Windows.

Note: *Unfortunately, OpenOffice does not have particularly good rtf import / export capabilities (it doesn't even round-trip correctly through itself!) and thus copying / pasting of lists between LiveCode and OpenOffice will not work reliably or correctly.*

plainText format

Paragraphs with a non-empty *listStyle* are appropriately exported in plain text form when using the *plainText*, *unicodePlainText*, *formattedText* and *unicodeFormattedText* properties.

For example, the above example would be rendered (for plainText) as:

        \<tab\>1.\<tab\>Numbered Item 1\<return\>

        \<tab\>2.\<tab\>Numbered Item 2\<return\>

        \<tab\>•\<tab\>Bulleted Item 1\<return\>

        \<tab\>•\<tab\>Bulleted Item 2\<return\>

Persistence

As it stands the *listStyle* property does not save into the stack-file and will not do so until version 5.0 when the file format is revised (for various technical reasons, it is not possible to add this as a saveable property with the current stackfile format).

It is recommended that htmlText be used to save the content of fields in custom properties, for restoration on reload.

Important: This feature is currently highly experimental. This means that it may not be complete, fail in some circumstances that you would expect it to work, or change considerably before becoming final. Additionally, it might have problems or gotchas that make it significantly harder to use than other LiveCode features at this time.

XML Namespace support (4.5.1)

By default, revXML will process XML Namespace related tags and attributes in a way that is inaccessible to revXML. This can cause problems when a script needs to process the namespaces itself.

To resolve this issue two additional functions mimicking the behavior of revCreateXMLTree and revCreateXMLTreeFromFile have been added to revXML. These functions are:

        revCreateXMLTreeWithNamespaces

        revCreateXMLTreeFromFileWithNamespaces

The difference between these and the originals is that these two functions ignore namespace tags and attributes, and instead return them as part of the tree without doing any processing.

For example, consider the following XML tree:

```
<db>
 <bb:record>
   <bb:firstname>Bugs</bb:firstname>
   <bb:lastname>Bunny</bb:lastname>
   <zz:zip>10101</zz:zip>
 </bb:record>
 <xx:record>
   <xx:firstname>Bugs</xx:firstname>
   <xx:lastname>Bunny</xx:lastname>
   <xx:zip>10101</xx:zip>
 </xx:record>
</db>
```

Will get interpreted as follows:

        revCreateXMLTree   **revCreateXMLTreeWithNamespaces**

```
db                                  db
  record[1]                           bb:record[1]
    record[1]/firstname                 bb:record[1]/bb:firstname
    record[1]/lastname                  bb:record[1]/bb:lastname
    record[1]/zip                       bb:record[1]/zz:zip
  record[2]                           xx:record[2]
    record[2]/firstname                 xx:record[2]/xx:firstname
    record[2]/lastname                  xx:record[2]/xx:lastname
    record[2]/zip                       xx:record[2]/xx:zip
```

The latter form is more appropriate in the case that an application needs the namespace information to perform correctly.

Runtime execution stack configuration (4.5.1 – experimental)

In order to be able to more reliably control the maximum level of recursion, a new global property **stackLimit** has been introduced.

This property allows a script to set (in bytes) the maximum size of the (runtime) stack the engine uses for recursive computation. A change in the setting will only take effect when all currently executing handlers complete, and at this time the stack size limit will be reconfigured to the given limit, or the nearest amount to it depending on available memory.

The stackLimit currently in effect can be fetched using **the effective stackLimit**.

The recursionLimit property is now bounded by the stackLimit – attempts to set the recursionLimit greater than the stackLimit will see it downwardly adjusted to the maximum current size allowed.

Note: The changes to the recursionLimit property and the new stackLimit property are only implemented on Windows at present.

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

revBrowser improvements (4.5.1 – experimental)

There is a long standing issue with revBrowser that causes browser instances to be lost whenever the stack it is attached to has its window re-created. Previously, cases where this would occur had to be avoided when a browser was present on a stack.

To resolve this problem a new property has been added to browser instances – windowId. The windowId property allows the stack to which a browser instance is attached to be changed after it has been created.

If the windowId is set to 0, the browser instance is temporary hidden. If the windowId is set to a valid stack windowId, the browser instance will move to that stack.

For example, to toggle the resizable property of a stack hosting a browser use the following code:

> revBrowserSet pBrowserId, "windowId", 0
>
> **set the** resizable **of stack** pBrowserStack **to** pNewResizeableValue
>
> revBrowserSet pBrowserId, "windowId", **the** windowId **of stack** pBrowserStack

Important: This feature is currently experimental. This means that it may not be complete, or

may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Non-blocking DNS resolution (4.5)

The **open socket** command no longer blocks on DNS resolution. Instead, if resolution is required the command will return immediately and the DNS lookup will happen in the background. If resolution fails, then a **socketError** message is sent in the same was as if connection fails.

For applications using **hostNameToAddress** directly, its syntax has been augmented:

> **hostnameToAddress(***hostname*, [*callback*])

If the *callback* parameter is specified then the call will return immediately and upon completion of the lookup, the callback will be invoked with the resolved address as a parameter.

Improved cursor support (4.5)

Cursor support has been improved in several ways.

The image will now automatically process any image to convert to a form suitable for display on the current platform and screen depth taking this burden off the developer. Specifically, the engine will scale the image down to the appropriate size, and reduce the number of colors to the appropriate number of colors.

Additionally, the engine has been updated to take advantage of support for larger and potentially alpha-blended cursors on platforms that support this. Specifically:

- Windows XP and above support full alpha-blended cursors up to 64x64

- More recent Linux distributions support alpha-blended cursors up to a size of 64x64

- Mac supports alpha-blended cursors up to 256x256

The engine now picks up the GTK cursor theme on Linux. *(4.5.1)*

Note: It appears that alpha-blending support depends on the current screen depth on some platforms.

Improved image export (4.5)

BMP Export

The export command can now produce Windows BMP format images in the same was at it previously did for gif and png:

> **export** *target* **as bmp**

Palette reduction

The **export** command can now perform color reduction. To support this, the following new forms have been added:

> **export** *target* **as (gif | png | bmp) with palette** *colors*

> **export** *target* **as (gif | png | bmp) with (standard | optimized) palette**

> **export** *target* **as (gif | png | bmp) with** *count* **color optimized palette**

The first form allows you to specify a list of up to 256 colors to use in the final palette.

The second form will use either the standard 'websafe' palette, or will compute an optimized palette with at most 256 colors.

The third form allows you to choose the size of the optimized palette to be generated. The number of colors can be at most 256.

If the image has any sort of transparency, then this will utilize one palette entry. (In particular if you ask for a 256 color palette with a transparent image, you may get only 255 colors).

In all cases, optional dithering will be performed as determined by its **dontDither** property.

Note: It is a (relatively) slow process to compute an optimized palette and then remap an image against it – it should not be considered a real-time operation except for very small images.

Note: There is no support for exporting a JPEG with a reduce palette as this format is for continuous-tone images and as such the notion of palette makes no sense.

Raw data export (experimental)

It is possible to export raw image data using the following forms:

> **export** *target* **as raw with palette** *colors*
>
> **export** *target* **as raw with (standard | optimized) palette**
>
> **export** *target* **as raw with** *count* **color optimized palette**
>
> **export** *target* **as raw [argb | bgra | abgr | rgab]**

The first three of these operate in the same way as for the other formats as described above except that instead of formatted image data you get the raw palette indices packed appropriately depending on the size of the palette:

> <= 2 colors will be 1 bpp
>
> <= 4 colors will be 2 bpp
>
> <= 16 colors will be 4 bpp
>
> <= 256 colors will be 8 bpp

The final form allows export of the full 32-bit data of the image with 8 bits per component. In this case, the components are not pre-multiplied with any alpha channel, and appear ordered in memory in increasing bytes.

e.g. The argb form will give you:

> byte 0 = alpha
>
> byte 1 = red
>
> byte 2 = green
>
> byte 3 = blue

Important: This feature (raw data export) is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Color profile support (4.5)

The engine will now attempt to use any embedded ICC color profile information that is present in JPEGs and PNGs.

If a JPEG contains a color profile then the engine will attempt to use it to translate the image's colors to the default screen color space before display.

When printing JPEGs, the original JPEG data will now be sent to the printer directly whenever possible, In particular, this means that any intermediate color matching for the screen that the engine performs does not affect the printed output.

As an additional side-effect of color profile support, the engine now also supports YCCK and CMYK JPEG images.

Note: *Support for color profiles depends on the OS - in particular Windows XP and earlier only support ICC v2 profiles, Vista onwards supports ICC v4.*

Print to pdf support (4.5)

The engine printing system has been augmented with the ability to *print to pdf*. This feature uses the existing print-loop model and virtually no code changes are needed to make existing code use it.

Print loop integration

To start a print loop that outputs directly to a pdf rather than the currently configured printer, use the following form:

> **open printing to pdf** *filename* **[with options** *optionArray* **]**

This should be performed *instead of* the usual **open printing** command.

Note: *It is not necessary (and unwise!) to set the **formatForPrinting** option on stacks which are being printed to PDF.*

Options

The options array which can be optionally specified when opening a print loop for pdf allows you to add entries to the resulting PDF's *Document Information Dictionary*.

The following keys are supported:

> Title, Author, Subject, Keywords, Creator, Producer

Their values can be any string.

Hyperlink support

When inside a pdf print loop you can use additional commands to define hyperlinks and their targets.

To define a target for an internal hyperlink, use the following form:

> **print anchor** *name* **at** *anchorPoint*

Here *name* is used to identify the anchor in related **print link** commands, and *anchorPoint* is the location on the current page to which any such link should jump to.

To define a hyperlink use the following form:

print link to *target* **with rect** *linkRectangle*

If *target* is a url, then the given rectangle will jump to that url when it is clicked. If *target* is not a url, it is assumed to be the name of an internal anchor as specified by the **print anchor** command and clicking in the given rectangle will jump there instead.

When printing fields, any text that has its **linkText** property set and has the textStyle **link** set will be treated as if a **print link** command had been executed with the contents of the property as target, and the formattedRect of the text as rectangle.

Runtime dependency

The implementation of pdf printing resides in a dynamic library called *revpdfprinter*. The standalone builder will automatically include this in the appropriate place when building standalones.

Any custom code that deals with standalones which use this feature must make sure that the revpdfprinter library resides next to the engine executable (On Mac, this is inside the Contents/MacOS folder, not next to the bundle).

Slave process improvements (4.5)

A number of issues with the *open process* command and the engine itself have, up until now, conspired to make it difficult (if not impossible!) to either run a slave process, or use the engine as slave on all platforms.

These issues have been resolved in this version, thus making it straightforward to run another process and poll for input and output over stdin/stdout.

The typical form for this is along the following lines (this example assumes the process being executed outputs whole lines):

```
command startSlave pProcess
      open process pProcess for text update
      send "monitorSlave pProcess" to me in 50 millisecs
end startSlave

command monitorSlave pProcess
      repeat forever
            # Loop until there are no more lines to read.
            read from process pProcess for 1 line in 0 millisecs
            if the result is empty then
                  # The slave has sent us something, so process it and loop for
                  # (potentially) more data.
            else if the result is "timeout" then
                  # There is nothing waiting for us, so exit repeat
                  exit repeat
            else if the result is "eof" then
                  # The slave has terminated, so do any final processing and finish
                  # monitoring.
                  close process pProcess
                  exit monitorSlave
            else
                  # Some error has occurred!
                  exit monitorSlave
            end if
      end repeat
      send "monitorSlave pProcess" to me in 50 millisecs
end monitorSlave
```

Videograbber improvements (4.5)

Some work has been done on storing and fetching settings from the videograbber.

On Windows and Mac, audio and video settings are now correctly retrieved and set using **revVideoGrabSettings** and **revSetVideoGrabSettings**.

On Windows (when using DirectShow), the 'Camara Control' settings are now saved and restored via the grab settings commands. Specifically: pan, tilt, roll, zoom, exposure, iris, focus and flash.

Elevated process support (4.5 – experimental)

Sometimes it is necessary to perform operations on the local machine as an administrator, and a typical pattern for a GUI application doing this is for it to prompt for authentication at certain points.

Modern operating systems do not permit a process to elevate itself, nor grant itself increased privilege. Instead, they only allow a running process to launch another process with increased privilege. Therefore, in order to support this, a new form of the **open process** command has been introduced that can launch a slave process with elevated permissions:

> **open elevated process** *process*s [**for [text | binary] (read | write | update | neither)]**

This form operates identically to the normal version, except that engine will ask the system to launch the given process with admin/root privileges.

The standard way for a GUI application that needs to perform privileged operations to be structured is to split the application into two parts: a GUI front-end that interacts with the user, and a command-line back-end that is run with elevated permissions. These two parts can then talk to each other using a standard master-slave approach, or some other form of IPC such as sockets.

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Status icon support (4.5 – experimental)

Windows, Linux and Mac all have an area where so-called 'status icons' can be displayed. On Windows this is the system tray on the bottom right of the start bar, on Linux this is typically the right of the panel at the top of the screen, and on Mac this is on the menubar.

The engine has support for adding a single status icon, and it can be configured using **the statusIcon**, **the statusIconTooltip** and **the statusIconMenu**:

> **set the statusIcon to** *imageId*
>
> **set the statusIconMenu to** *iconMenuSpec*
>
> **set the statusIconToolTip to** *toolTip*

Here *imageId* is the id of the image you wish to use as the icon. It will be scaled down automatically to the appropriate size for the platform and then set. The *toolTip* specifies what message appears when the user hovers over the status icon.

The *iconMenuSpec* allows you to configure a menu that will appear when the user does a 'menu' click on the icon. This string uses a subset of the standard engine menu specification:

[<tab> *] ['('] <label> ['|' <tag>]

Here the number of tabs determines the depth of the menu (i.e. use this to create sub-menus). The optional tag is used when calling the **statusIconMenuPick** message.

Before the engine displays the status icon menu, it will send a **statusIconMenuOpening** menu to the current card of the defaultStack. You can use this opportunity to change the icon menu before it is displayed, this is an analog to handling *mouseDown* in a menu button.

When the user selects an item from the dock menu, the engine will send an **statusIconMenuPick** message to the current card of the default stack:

iconMenuPick *which*

Here *which* will be a list of labels or tags (if specified) separated by '|' which determines which item was selected.

In addition, the engine will send the following message in response to clicks on the icon:

statusIconMenuClick *button*

statusIconMenuDoubleClick *button*

You can use these to perform an appropriate action.

Note: If you wish to display a menu from the status icon you must use the statusIconMenu property, attempting to open a normal popup menu in response to one of the click messages is not guaranteed to work.

*Note: This syntax is only implemented on Windows at the moment and replaces the previously unsupported use of **the icon** and **the iconMenu** for this purpose. The properties specified above will have no effect on Mac and Linux at this time.*

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Listing sub-keys in the registry (4.5 – experimental)

To get a list of sub-keys in the Windows registry use the following function:

listRegistry(*parentKey*)

This will return a return-delimited list of sub-keys, i.e. those keys which are direct children of the given *parentKey*. The specified key should be in the same format as the other registry functions.

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

HTTPS – automatic root certificate discovery (4.5 – experimental)

In previous versions it was necessary to set the *sslCertificates* property to the root certificates that HTTPS connections should be verified against. Support has now been added to locate and load the root certificates installed (and kept up to date) as part of the OS.

This uses the standard root certificate keychain on Mac, the standard root certificate store on Windows and uses a number of heuristics to locate this information on Linux.

You can easily find out if the system-installed root certificates are being found by running the following command in the message box:

get url "https://www.google.com"

put the result & **return** & **it**

If this results in an error about verification failure then it is likely that root certificates have not been found. Please let us know (particularly on Linux) if you find this simple test fails, making sure you give us full details of your system (e.g. Linux distribution and version).

*Note: Unfortunately this feature does not currently work correctly on Mac 10.6.x. For now, we advise including an appropriate root certificates collection with your application, as was previously necessary, and setting the **sslCertificates** property appropriately.*

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Mac – dock icon support (4.5 – experimental)

Previously unsupported syntax for manipulating the dock icon on Mac is now experimental.

Choosing an image

The current dock icon image can be set by using the global **icon** property:

set the icon to *imageId*

The engine will attempt to find an image with the given id, resize it to 128x128 and then set it as the dock icon for the application.

This property has no effect on other platforms.

Note: The image is only guaranteed to persist while the application runs, although in some cases the OS does appear to cache it beyond this.

Configuring the dock icon menu

In addition to changing the dock icon image, you can also configure the menu that appears when the user clicks on it.

To set the dock icon menu use the global **iconMenu** property:

set the iconMenu to *iconMenuSpec*

Here, *iconMenuSpec* is a string describing the menu. This uses a subset of the standard menu specification syntax. The string should be a return-delimited list of items specified as follows:

[<tab> *] ['('] <label> ['|' <tag>]

Here the number of tabs determines the depth of the menu (i.e. use this to create sub-menus). The optional tag is used when calling the **iconMenuPick** message.

Before the engine displays the icon menu, it will send a **iconMenuOpening** menu to the current card of the defaultStack. You can use this opportunity to change the icon menu before it is displayed, this is an analog to handling *mouseDown* in a menu button.

When the user selects an item from the dock menu, the engine will send an **iconMenuPick** message to the current card of the default stack:

> **iconMenuPick** *which*

Here *which* will be a list of labels or tags (if specified) separated by '|' which determines which item was selected.

Important: This feature is currently experimental. This means that it may not be complete, or may fail in some circumstances that you would expect it to work. Please do not be afraid to try it out as we need feedback to develop it further.

Maintenance

Noteworthy changes

The following bug fixes or changes are ones which may have an impact on existing code and/or may be important to a wide audience.

(changes specific to the current build are highlighted with a gray background)

Mac OS 10.3.9 is no longer supported

Support has been dropped for Mac OS 10.3.9.

Clipboard format for styled-text on Mac now RTF

The styled text format that LiveCode exports to the clipboard on Mac has been updated to be RTF – this brings it inline with both Windows and Linux, and will allow support for hyperlinks and listStyles to be supported in a subsequent build.

SQLite updated to 3.7.2 with Full Text Search enabled

The verison of SQLite compiled into dbSqlite has been updated to version 3.7.4. In addition, the 'FTS' feature has been compiled in. For more information about this version of SQLite and the FTS feature please see the documentation at *http://www.sqlite.org*.

Non-character keys can cause erroneous keyDown messages on Mac

Previously keys such as page-up, page-down and the function keys would cause keyDown messages to be sent to a field but with a nonsensical parameter. These keys no longer generate keyDown in line with other platforms.

The listStyle property has better RTF import / export support

Support for both types of list specification in RTF has been added, increasing interoperability with other applications when using the clipboard. See the section on the listStyle feature for more information.

Bug 818 – About item always present and enabled on Mac

The engine will now only add an *About* item to the Apple menu if it finds a menu with name *Help* whose last item has tag *About* or label beginning with *About.*

Furthermore, the about item is enabled only if the found item is enabled and the help menu is enabled.

Bug 3071 – Preferences item cannot be disabled on Mac

The engine will now only enable the Apple menu preferences item if the following are all true:

- There is a menu with name *Edit*
- The edit menu is enabled
- The last item of the edit menu has tag *Preferences*, or has label beginning with *Preferences*
- The preferences item is enabled

Bug 1197 – backgroundBehavior policy now enforced

The engine now strictly enforces the following two rules:

- a nested group cannot have the backgroundBehavior set to true
- a group placed onto multiple cards cannot have its backgroundBehavior set to false

On loading stacks in the IDE, a consistency check will be performed which will result in these rules holding after load.

Bug 2122 – Styled text overruns tab-stops

You can now use styled text correctly when a field has the *vGrid* property turned on. The text will be clipped correctly at cell boundaries.

Bug 4026 – importing images > 4091 wide causes corruption on Mac

The long standing limit on the width of image objects on Mac has been lifted. Images are now limited by available memory, rather than the width.

Bug 7334 – rtfText output is now effective

The rtfText property is now 'effective' in the sense that it contains styling for all characters, including those which inherit text properties from the field (or more distant ancestor). This change has been made to make it more useful for clipboard export. If you wish to fetch the contents of the field with the inheritance structure intact, use the htmlText instead.

Bug 9301 – menuHistory now has consistent behavior for all menu button types

Previously, setting the menuHistory would only work reliably for option/combo buttons. This has been rectified and setting the menuHistory (or using select menuitem … of menu …) will now cause the correct *menuPick* message to be dispatched for all menu button types.

Note that for option and combo buttons, such a message will only be sent if the menuHistory changes, this is consistent with these objects having a notion of 'currently selected item'.

Bug 9378 – 'mailto:' URLs not recognized by 'print link to'

In order to better support usage, the syntax and functionality of the 'print link' command has changed. In particular, the print link command now accepts a 'url' adjective to better specify the type of link. This will require changes to existing scripts in a very specific case.

Code such as:

> **print link to url** "file:mylinktext.txt" **with rect** tMyLinkRect

Needs to be changed to:

> **print link to** (**url** "file:mylinktext.txt") **with rect** tMyLinkRect

Bug 9412 – Messages passing through a background are received twice if passed

The message path order for backgrounds has been revised slightly (in a backwards-compatible fashion) to take proper account of passing of messages. For a control within a background, the message path is the following:

> target
>
> target's owning groups
>
> target's background
>
> card
>
> card's backgrounds not including the target's background
>
> stack

The path for a message sent to a background is:

> target background
>
> card
>
> card's backgrounds not including the target's background
>
> stack

Note: The above order effectively means that a background's script is only behind the card for controls it does not contain which retains compatibility with previous interpretations of backgroundBehavior.

Specific bug fixes (4.6.1)

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

	revCaptureSetRecordFrameRate throws an error
9483	the tempName can generate non-unique file paths
9486	Delay after dropping on Mac
9489	Importing 2 color PBMs causes a crash on save
9503	Formatted width of listStyle paragraphs is incorrect
9504	Focus border remains if focused field disabled by script

Specific bug fixes (4.6)

> Crash when setting the 'id' of a template object
> Wrong drag-drop result returned to OS in some cases (causing incorrect snap-back)
> Crash when image fails to load due to lack of memory
> Underline/strikethrough not displayed in fields in most cases
> 'repeat for each key' does not preserve loop var value on 'exit repeat'
> 'print bookmark' doesn't work correctly with non-ASCII titles
> The 'formattedText' includes extra linebreaks
> Memory leak when displaying some revOnline dialogs.
> Incorrect htmlText output for listStyle for last paragraph in field
> Unable to fetch 'the listStyle' from field chunks
> Border (incorrectly) shows around PDF hyperlinks in Preview
> Intermittent problem with text styling when printing as PDF
> Crash when cutting card containing a non-background shared group

Copying from dictionary into TextEdit on Mac fails sometimes
Transient parts of objects (e.g. focus borders) do not appear in graphic effects

 in htmlText can cause spurious listStyles to be applied
<p> tags are not optional when in <li> tags
'go as sheet' doesn't disable the parent window on Windows
'ask folder' uses new style dialog on Vista and later
Fonts in styled text can paste with incorrect style on Mac in some cases
Non-character keys can cause erroneous keyDown messages on Mac
Text linkClicked parameter contains an incorrect trailing character in some cases

818	About item always present and enabled on Mac
1197	Correct setting of backgroundBehavior now enforced.
2122	Styled text overruns tab-stops
3071	Preferences item cannot be disabled on Mac
4026	Importing images > 4091 wide causes corruption on Mac
5141	Menus with unicode titles sometimes disappear on non-English systems
5869	Fields do not use standard Mac text editing shortcuts
7102	Oracle driver not available for Intel
7334	'the rtfText' output is not effective (i.e. does not contain styling for inherited runs)
7358	Problems with direction specifiers in visual effects
7974	'ask file' works incorrectly in some cirumstances on Mac
8612	revBrowserRefresh breaks subsequent searches on Windows
8790	Driver name should be lower-cased before loading
8991	Pending messages not dispatched until mouse moved after drag-drop action
9192	'record sound' does not create a file properly if given a relative path
9282	'lock messages' not reset after modal dialogs (including answer and ask commands)
~~9284~~	~~Double-clicking on a word in a locked field does not select it~~
9288	'the properties' does not take account of an explicit card reference
9293	Unicode text causes incorrect table alignment
9298	Large linesizes of graphics cause incorrect hit-testing
9301	'the menuHistory' has inconsistent behavior for pulldown/popup/cascade menu buttons
9308	Ids of controls can only be changes on the current card and crashes otherwise
9311	Keys of options passed to revPDFPrinter are not case-insensitive
9332	'launch url' doesn't always return to LiveCode
9352	The menuHistory of some menu button types not set correctly
9353	Rectangle graphics of height 1 don't print to PDF
9355	Export snapshot confuses focus of fields
9359	Can't quit when revBrowser has focus on Mac
9368	CMYK JPEGs don't appear in printed PDFs
9375	The 'templateField' does not convert text to htmlText
9377	mouseDoubleUp not sent to locked and list fields
9378	'mailto:' URLs not recognized by 'print link to'
9380	listStyle doesn't report 'mixed' when it should in many cases
9381	'export snapshot' does not render anti-aliased text correctly on Windows
9382	Radio buttons with custom icons don't update correctly when in backgounds
9383	Crash when setting 'the backgroundBehavior' of the templateGroup
9384	Unnecessary limit on the number of menu buttons on open cards on Mac
9388	Change in how htmlText is interpreted
9395	Crash when setting field rect on Linux
9396	Tooltips don't have drop-shadows on Windows 7

9399   Can't get the short name of the dragSource
9400   'the text of the selectedLine' is always empty
9403   Right-click then Cut on Mac causes crash on empty image
9404   The formattedHeight of a field counts the scrollbarWidth twice
9410   Current external load process is dependent on current folder on Windows
9412   Messages passing through a background are received twice if passed
9413   The ctrlKey synonym is tied to commandKey
9417   Only the first link on a line ever activates when listBehavior is true
9419   The pixel properties only work if the object is on the current card of an open stack
9428   Modal windows do not behave correctly on Linux
9435   BMP images with depth < 32 and odd width do not import correctly
9439   Text does not render correctly on non-primary screen on Linux (not tested)
9447   The linkClicked message does not pass the full linkText value as a parameter
9451   Numeric keypad input does not work on Mac
9466   Tab key does not insert a tab into fields.

IDE Changes

Getting folder locations within the IDE

If you write plugins, or have code that relies on the location of IDE files then please ensure you use the following access functions to locate them:

revEnvironmentToolsPath()	The location containing the main IDE files.
revEnvironmentToolsetPath()	The location of the main IDE stacks.
revEnvironmentExternalsPath()	The location of the externals that come with the IDE.
revEnvironmentPluginsPath()	The location of the plugins that come with the IDE.
revEnvironmentRuntimePath()	The location of the standalones that come with the IDE.
revEnvironmentDocumentationPath()	The location of the documentation files.
revEnvironmentResourcesPath()	The location of the resources that come with the IDE.
revEnvironmentCustomizationPath()	The location of the IDE customization folder.
revEnvironmentUserCachePath()	The location of the folder to use for caching files.
revEnvironmentUserPreferencesPath()	The location of the folder to use for preference files.
revEnvironmentUserExternalsPath()	The location of the folder to use for additional externals.
revEnvironmentUserPluginsPath()	The location of the folder to use for additional plugins.
revEnvironmentUserResourcesPath()	The location of the folder to use for additional resources.

Important: Third-party IDE extensions must avoid placing any files inside the application bundle or under revEnvironmentToolsPath() (not least because you will probably not have privileges to do so!). Instead, they should use the user-externals and user-plugins paths as provided. These paths are determined by the user's customization path setting, configurable in the preferences.

Update checker

The update checker and its corresponding menu item in *Help* have been removed. The version checking method is changing, and this option will re-appear in a subsequent release.

Affiliate registration

The affiliate registration item in the *Help* menu has been removed The method of affiliate registration is changing, and this option will re-appear in a subsequent release.

Database query builder

Use of the database query builder has been deprecated.

To access its functionality, choose the *Enable database query builder* option on the *Compatibility & Updates* pane of the preferences.

Note: The database query builder will be removed entirely in the next or subsequent non-maintenance release.

Android Support (4.6.1)

The IDE now contains built-in support for testing and deploying Android applications. The

following features have been added to support this:

- An Android pane has been added to the Standalone Builder

- *Save as Standalone Application* can now build Android applications

- The *Test* button supports both Android and iOS application testing.

- Configuration of the test target (device or emulator) via a *Test Target* submenu in the *Development* menu.

- Configuration of SDK location for Android in the *Mobile Support* pane of the Preferences.

More details on how to use these features can be found in the *Android Release Notes* accessible via the *Help* menu.

Note: If you do not have an Android deployment pack, you can still use the Android feature in trial mode. In this case the applications you build will contain a forced banner on startup lasting 5 seconds, and they will quit after one minute.

iOS Support (4.5.2)

The IDE now contains built-in support for testing and deploying iOS applications. The following features have been added to support this:

- An iOS pane has been added to the Standalone Builder

- *Save as Standalone Application* can now build iOS applications

- A *Test* button has been added to the menubar to enable fast and easy access to running applications in the simulator

- Configuration of simulator version and type can be accessed via a *Test Target* submenu in the *Development* menu.

- A *Mobile Support* pane has been added to the preferences to enable configuration of iOS SDKs (needed by deployment and simulation).

More details on how to use these features can be found in the *iOS Release Notes* accessible via the *Help* menu.

Note: If you do not have an iOS deployment pack, you can still use the iOS feature in trial mode. In this case the applications you build will contain a forced banner on startup lasting 5 seconds, and they will quit after one minute.

Access to resources folder (4.5.2)

The IDE comes with a Resources folder containing the sample projects, example stacks and other miscellany.

Previously this was easily accessible on all platforms by navigation to the installed IDE folder. As this folder is now (on some platforms) inside an application bundle, a new item in the Help menu **Example Stacks and Resources** has been added to open up the folder in the file manager on the running system.

Standalone builder

Windows – UAC Manifest

It is now possible to specify what action UAC should take on Windows Vista and higher when the standalone is launched. You can choose one of the following options:

Default	No UAC option is provided in the manifest.
Save as Invoker	The application will run with the same privileges as the process that invoked it.
Highest available	The application will be elevated to the highest privilege level the current user is allowed.
Require administrator	The application will be run as administrator after prompting the user for appropriate login/elevation rights.

Web

Using the 4.5 standalone builder to build for Web will create revlets. These are compatible with the existing revWeb plug-in available from revweb.runrev.com.

An updated version of the IDE for producing LiveCode Applets and the associated LiveCode player will be made available in due course.

Note: As the currently available revWeb plug-in uses the 4.0 engine, you must be careful to only use features that are present in that version.

Android (4.6.1)

There is now a pane for configuring Android applications in the Standalone Builder.

More details on how to use the Android settings pane can be found in the 'Android Release Notes' accessible via the Help menu.

iOS (4.5.2)

There is now a pane for configuring iOS applications in the Standalone Builder. At this time, iOS standalone building has the following limitations (compared to desktop/web deployment);

- A stack configured for iOS cannot also deploy to other non-mobile platforms

- The *Copy Referenced Files* feature is not implemented for iOS builds.

- The *Inclusions* and *Property Profiles* feature is not implemented for iOS builds.

- The *Stacks* configuration options are not available for iOS builds.

More details on how to use the iOS settings pane can be found in the 'iOS Release Notes', accessible via the Help menu.

Datagrid

The datagrid is currently at version 1.0.2 (build 13).

1.0.2 (build 12)   Added specific scrollbar width for linux platform.

DeleteFieldEditorAndOpenNext now continues looking for a column to edit if EditValue is passed in the behavior script. This allows the developer to skip a column for editing. Previously this would only occur if EditValue was not handled.

DeleteFieldEditorAndOpenNext now skips invisible columns.

The control that DeleteFieldEditorAndOpenNext was targeting would sometimes be changed behind the scenes. This would cause CloseFieldEditor to be sent to the wrong control.

1.0.2 (build 13)   ResetList code is now in dgResetList. This allows the developer to intercept ResetList and manually call dgResetList while adding their own logic.

Maintenance

Noteworthy changes

The following bug fixes are ones which may have an impact on existing code and/or may be important to a wide audience.

(changes specific to the current build are highlighted with a gray background)

Minor usability improvements

After observing a number of users using LiveCode for the first time, the following minor changes have been made:

- The default state of the paint tools in the tools palette is opened so they are visible on start up (existing installs are unaffected).

- The 'Hide Palettes when editing scripts' and 'Hide Message Box when editing scripts' options are now defaulted to false (existing installs are unaffected).

- On the menubar the 'Rev Online' button has been renamed 'User Samples' and a new 'Tutorials' button introduced linking directly to http://lessons.runrev.com.

Specific bug fixes (4.6.1)

(bug fixes specific to the current build are highlighted in bold)

	Building a standalone with oracle causes a warning on Mac Intel
	Building an Android standalone in non-Commercial mode causes an error
9470	**Chrome sometimes sets global environment variables that break the message box**
	Simulate button acts on topStack, rather than the mainStack of the topStack
	Dictionary pane in script editor sometimes displays xmlerr.
8975	Start center recent files can become confused
9491	Unicode text does not display which font in Text menu
9496	Inspector chokes with stack names containing "!"
9520	HTTP headers are split by word, rather than ':'

Specific bug fixes (4.6)

Added shortcut for 'Paste Unformatted'

Mac field shortcuts do not work correctly in the Script Editor
The dictionary does not open correctly in some cases.
Pressing option in some IDE fields cause them to revert the field.
New datagrids are not created properly and can't have their Row Template's shown.
LiveTalk still present in the first pane of the preferences GUI.
Simulator menu does not filter list by available engines

4864	Property inspector's treatment of link style properties does not reflect reality
7250	'edit custom prop name' dialog resizable when it shouldn't be
8576	IDE processes openDoc filename incorrectly
8775	Continuations have an inconsistent indent in the S/E
9291	Body text in the about dialog is clipped initially
9292	Simulate button does not have a tooltip
9299	Substacks are unsorted in the app browser
9316	CFBundleVersion not set when building Mac standalones
9453	Dictionary note rating broken

Resource and Documentation Changes

User Guide

No changes.

Dictionary

A substantial number of dictionary entries have had minor tweaks and fixes applied in this release, most as a result of bug reports in the quality control center. Additionally, new syntax entries have been added for much of the new syntax added since 4.0.

Mobile Examples

The *Mobile Examples* are a collection of stacks demonstrating various features specific to the mobile platforms LiveCode supports.

These are accessible from *Help > Example Stacks and Resources* and are within the *Mobile Examples* folder.

The following changes have been made in this release:

Sound Example

The example queue has been updated to use the changed functionality of *iphonePlaySoundOnChannel* in *next* mode when there is no sound playing (i.e. to prepare the sound to ensure it plays with zero latency when needed).

To support this change, a *Pause* and *Resume* button have been added mapping to *iphonePauseSoundOnChannel* and *iphoneResumeSoundOnChannel*.

Previous Release Notes

4.5.0 Release Notes http://www.runrev.com/downloads/livecode/4_5_0/LiveCodeNotes-4_5_0.pdf

4.5.1 Release Notes http://www.runrev.com/downloads/livecode/4_5_1/LiveCodeNotes-4_5_1.pdf

4.5.2 Release Notes http://www.runrev.com/downloads/livecode/4_5_2/LiveCodeNotes-4_5_2.pdf

4.5.3 Release Notes http://www.runrev.com/downloads/livecode/4_5_3/LiveCodeNotes-4_5_3.pdf

Revisions

Revision 1	MW	Document created for issue with 4.6.0-dp-1.
Revision 2	MW	Updated ide bug fix list for 4.6.0-dp-2.
		Updated engine bug fix list for 4.6.0-dp-2.
		Added section on 'plain text' field properties
		Updated listStyle property section
		Updated noteworthy engine changes section
Revision 3	MW	Updated ide bug fix list for 4.6.0-dp-3
		Updated engine bug fix list for 4.6.0-dp-3
		Added section on print to pdf document outline support
Revision 4	MW	Updated ide bug fix list for 4.6.0-dp-4
		Updated engine bug fix list for 4.6.0-dp-4
		Added section on 'export snapshot from … with effects'
		Clarified section on print to pdf with regards to link printing
Revision 5	MW	Updated engine bug fix list for 4.6.0-dp-5
		Added section on new print link syntax
Revision 6	MW	Updated engine bug fix list for 4.6.0-dp-6
Revision 7	MW	Updated engine bug fix list for 4.6.0-dp-7
		Updated datagrid section to mention update to 1.0.2 build 12.
		Added new 'Resource and Documentation Changes' section.
Revision 8	MW	Updated engine bug fix list for 4.6.0-dp-8
		Updated noteworthy engine changes section
		Updated listStyle rtf export section
Revision 9	MW	Updated engine bug fix list for 4.6.0-rc-1
		Updated dictionary section of resource and documentation changes.
Revision 10	MW	Updated ide bug fix list for 4.6.0-rc-2
		Updated datagrid section to mention update to 1.0.2 build 13.
Revision 11	MW	Updated ide bug fix list for 4.6.0-rc-3
Revision 12	MW	No changes.
Revision 13	MW	No changes.
Revision 14	MW	Updated engine bug fix list for 4.6.0-gm-2
Revision 15	MW	Updated engine bug fix list for 4.6.1-dp-1
		Updated ide bug fix list for 4.6.1-dp-1
		Added details of 'initially' clause to print bookmark section.
		Added section of new 'revCapture' API
Revision 16	MW	Updated engine bug fix list for 4.6.1-dp-2
Revision 17	MW	Updated engine bug fix list for 4.6.1-rc-1

Updated ide bug fix list for 4.6.1-rc-1
Added sections on Android support.
Revision 18   MW   Updated for 4.6.1-gm-1.
Revision 19   MW   Updated ide bug fix list for 4.6.1-gm-2.