LiveCode 6.0.1-rc-1 Release Notes

Table of Contents

Overview

LiveCode 6.0.1 is an update to 6.0 including various minor bug fixes and updates.

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.

For information about improvements made to the Android port of the engine, please see the Android 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
- 10.8.x (Mountain Lion) on Intel

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

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 6.0.1-rc-1
- Linux: /opt/runrev/livecode-6.0.1-rc-1
- Mac: /Applications/ LiveCode 6.0.1-rc-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 6.0.1-rc-1
- Linux: ~/.runrev/components/livecode-6.0.1-rc-1
- Mac: ~/Applications/ LiveCode 6.0.1-rc-1app

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-6.0.1-rc-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 licensing system ties your product licenses to a customer account system, meaning that you no longer have to worry about finding a license key after installing a new copy of LiveCode. Instead, you simply have to enter your email address and password that has been registered with our customer account system and your license key will be retrieved automatically.

Alternatively it is possible to activate the product via the use of a specially encrypted license file. These will be available for download from the customer center after logging into your account. This method will allow the product to be installed on machines that do not have access to the internet.

Multi-user and network install support (4.5.3)

In order to better support institutions needing to both deploy the IDE to many machines and to license them for all users on a given machine, a number of facilities have been added which are accessible by using the command-line.

Note: *These features are intended for use by IT administrators for the purposes of deploying LiveCode in multi-user situations. They are not supported for general use.*

Command-line installation

It is possible to invoke the installer from the command-line on both Mac and Windows. When invoked in this fashion, no GUI will be displayed, configuration being supplied by arguments passed to the installer.

On both platforms, the command is of the following form:

<exe> install noui *options*

Here *options* is optional and consists of one or more of the following:

-allusers	Install the IDE for all users. If not specified, the install will be done for the current user only.
-desktopshortcut	Place a shortcut on the Desktop (Windows-only)
-startmenu	Place shortcuts in the Start Menu (Windows-only)
-location *location*	The location to install into. If not specified, the location defaults to those described in the *Layout* section above.
-log *logfile*	A file to place a log of all actions in. If not specified, no log is generated.

Note that the command-line variant of the installer does not do any authentication. Thus, if you wish to install to an admin-only location you will need to be running as administrator before

executing the command.

As the installer is actually a GUI application, it needs to be run slightly differently from other command-line programs.

In what follows <installerexe> should be replaced with the path of the installer executable or app (inside the DMG) that has been downloaded.

On Windows, you need to do:

start /wait <installerexe> install noui *options*

On Mac, you need to do:

"<installerexe>/Contents/MacOS/installer" install noui *options*

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

Command-line activation

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

On both platforms, the command is of the form:

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

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

This action can be undone using the following command:

<exe> deactivate

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

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

On Windows, you need to do:

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

start /wait <livecodeexe> deactivate

On Mac, you need to do:

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

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

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

Engine Changes

Script Security changes (6.0 RC4)

We've refactored script security out of the main engine into a module in preparation for dual licence support.

We've updated the hash used by the security module. Stacks that have their password set in RC4 will not be unlockable in previous versions.

Image handling changes (6.0 DP 5)

The engine now has an image cache which is used to cache all decompressed image data. The size of the cache can be set using the global '*imageCacheLimit*' property, and the amount of data in the cache is returned by the global 'imageCacheUsage' property.

The image cache operates in a least-recently-used manner - if an image needs to be decompressed and there is not enough room, the images that were used longest ago are discarded from cache until there is enough room.

If an image has '*alwaysBuffer*' set to true, then it will be decompressed into the cache on card open. Images are processed from lowest layer to highest, so if too many images have alwaysBuffer on a card, images on lower layers will have their data discarded from the cache before ones higher up.

An image can be forced to be cached by using the new form of prepare:

prepare image <name> [**of** ...]

prepare image file <filename>

The image cache is keyed on absolute filename of image thus it is now highly efficient to use image objects referencing the same filename, rather than buttons referencing icons. In particular, there will only ever be one decompressed set of image data for a given (absolute) filename in the cache at any one time.

Note that rotated/scaled images cache the transformed image data in the cache also, but images will not share this transformed data.

Update to “is” and “=” operators (6.0 DP 3)

When comparing arrays with other values, the engine will now no longer coerce an array to empty when comparing in string/number context.

From 6.0 DP 3 onwards, expressions such as 'tArray **is empty**' will be true if and only if tArray contains the empty string.

For example:

put empty into tArray

answer tArray **is empty** -- *true*

put 100 **into** tArray["foo"]

answer tArray **is empty** -- *false (prior to 6.0 this would be 'true')*

Field Updates (6.0 DP 2)

Non-BMP unicode characters

Support has been added for non-BMP unicode characters in **htmlText** (previously they wouldn't convert properly).

Hexadecimal HTML entity references

Support has been added for hexadecimal entity references in **htmlText** (&#x...;)

HTML UTF-8 meta tag

Added support for 'meta' tag in htmlText - if the charset or content attribute contains utf-8 the string will be treated as UTF-8 (i.e. you can use a UTF-8 encoded **htmlText**, rather than having to use numeric entity references for unicode chars).

Strong and em HTML tags

Support has been added for 'strong' and 'em' tags in htmlText.

New listIndex paragraph property

A new paragraph property **listIndex** has been added. This can be used to set the index for the paragraph when a (ordered) listStyle is set. This translates to **htmlText** via the 'value' attribute on <li> tags.

New metadata paragraph property

A new paragraph property **metadata** has been added. Like metadata at the char level, you can now '**set the metadata of line** ... **of** ...' and it will be attached to the paragraph, a level above the char properties.

Note that this means if you want to set the metadata on the char level for a whole line you must do '**set the metadata of char** 1 to -1 **of line** .. **of** ...', rather than just '**of line**'.

ID caching (6.0 DP 2)

The engine now caches lookups of control references of the form “card id ...” and “control id ...”. This speeds up any access using those forms after the first time, turning an operation that would be completed in linear time into one that will be completed in constant time.

Note: This was originally mistakenly documented as added in 6.0 DP 1.

New global property – allowDatagramBroadcasts (6.0 DP 2)

A new global property **allowDatagramBroadcasts** has been added. This needs to be set to true before sending to a broadcast address (enables the per-socket flag for whether they are allowed).

See bug 10516.

Before and after handlers (6.0 DP 1)

Two new handler types have been added, **before** and **after.** Before and after handlers are exclusive to behavior scripts and are sent to the behavior script before and after all messages.

For example, consider a mouseDown message moving through the message path. It gets to an object with a behavior script:

1. The engine looks at the behavior script of the target object - If a **before** mouseDown handler is present, it executes it.
2. The engine next looks at the object script - If an **on** mouseDown handler is present, it executes it.
3. The engine now looks at the behavior script - If an **after** mouseDown handler is present, it executes it.
4. The engine finally looks at the object script - if a **pass** mouseDown or no mouseDown handler was present, it moves on to the parent object.

Before and after handlers allow developers to produce behavior scripts which can handle messages sent to a control without having any effect on the message path (unlike front and back scripts).

Relayer command (6.0 DP 1)

A new **relayer** command has been added in order to aid the manipulation of layers.

relayer *control* **(before | after) layer** *index*

relayer *control* **(before | after) control** *target*

relayer *control* **(before | after) owner**

relayer *control* **to (front | back) of layer i***ndex*

relayer *control* **to (front | back) of control** *target*

relayer *control* **to (front | back) of owner**

All forms work in a regular way - a control is moved relative to a target in one of four ways:

- *before* - the control's owner becomes the target's owner and is inserted before the target; the layer of the control becomes the layer of the target and the target and all subsequent controls shift up one).
- *after* - the control's owner becomes the target's owner and is inserted after the target; the layer of the control becomes one greater than the layer of the last child of the target and all subsequent controls shift up one.
- *front* - the control's owner becomes the target's owner and is inserted after the last child of the target (only valid if *target* is a container).
- *back* - the control's owner becomes the target's owner and is inserted before the first child of the target (only valid if *target* is a container).

The target is resolved in one of three ways:

- *layer* - the target control is taken to be the control with layer *index*.
- *control* - the target control is taken to be the given control.
- *owner* - the target control is taken to be the owner of the control.

The *control* parameter determines the control that is to be moved.

Note that the **relayer** command can only be used to move controls within a card - attempts to relayer controls from one card (or another stack) to another will throw an error.

The **relayer** command can throw a number of errors:

- 'relayer: couldn't resolve target object' - the target chunk is an invalid control or card
- 'relayer: couldn't resolve source control' - the source chunk is an invalid control
- 'relayer: source not a control' - the source chunk resolves to a non-control object
- 'relayer: target not a container' - the target chunk resolves to a non-container object (i.e. not a card or a group)
- 'relayer: source and target not on the same card' - the source control does not reside on target (if it's a card), or target's card
- 'relayer: layer not an integer' - the expression passed to the 'layer' form is not a valid integer
- 'relayer: bad layer expression' - an error was thrown while evaluating the layer expression
- 'relayer: target not a control' - in the before or after form, the target chunk resolves a non-control object
- 'relayer: cannot move a control into a descendent' - an attempt has been made to move a group into a child of itself.
- 'relayer: required objects disappeared while processing' - the target object and/or the source object were deleted in the process of relaying (this can happen when moving controls in and out of groups as messages might be sent to ensure focus is correct).

Effective rect of stack (6.0 DP 1)

The **effective rect** property of stacks now returns the rect of the given stack with its decorations and frame taken into account. The **effective rect** of a stack can also be set. Here, the rect of the frame of the stack will be set appropriately before setting the rect of the stack.

Import/export snapshot at size (6.0 DP 1)

New variants of the import snapshot and export snapshot commands have been added:

import snapshot … **at size** *width*, *height*

export snapshot … **at size** *width, height*

Use the “at size” extensions if you wish the engine to resize the snapshot taken to the dimensions specified.

Control at location (6.0 DP 1)

You can now fetch the control at a given location using two new functions:

controlAtLoc(*loc***)**

controlAtScreenLoc(*loc***)**

ControlAtLoc returns the control at the given location, with the location being relative to the current stack.

ControlAtScreenLoc returns the control at the given location, with the location being relative to the screen. **ControlAtScreenLoc** is currently only available on Mac and Windows.

Do in caller (6.0 DP 1)

A new variant of the do command has been added:

do *statementList* **in caller**

This will have the effect of executing the statementList in the context of the calling handler.

Split/combine as set (6.0 DP 1)

New variants of the split and combine commands have been added:

split *variable* **{ by | with | using }** *delimiter* **as set**

combine *variable* **{ by | with | using }** *delimiter* **as set**

The "**split** *tList* **with** return **as set**" command converts the passed variable to an array with the keys being equal to the original list and the values being true.

The "combine as set" command rebuilds the list using the delimiter passed. The values of the array are ignored.

Object selection started/ended (6.0 DP 1)

Two new IDE messages**, objectSelectionStarted** and **objectSelectionEnded,** have been added. The messages are sent to the current card. **ObjectSelectionStarted** is sent when in edit mode, the user drags over a control. **ObjectSelectionEnded** is sent when the drag action is complete.

Explicit line-breaks in fields *(5.5 DP2 – experimental)*

The engine will now interpret a numToChar(11) character in a field paragraph as an explicit line-break when the (effective) dontWrap of the paragraph is false. This allows multiple 'lines' to be displayed within a single paragraph.

The formattedText property has been updated to map any explicit line breaks to newlines.

Note that since the vGrid property turns dontWrap on for the paragraph, using the line-break char in table paragraphs will have no effect.

textStyle Property Array Variant (5.0.2 – experimental)

A new array variant of the **textStyle** property has been added allowing for access to individual

textStyles independently from the others by specifying the required style as an array key.

the textStyle[<style>]

Here, style can be one of bold, condensed, expanded, italic, oblique, box, threedbox, underline, strikeout, link.

Important: The features described above are 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 – 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.*

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 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.

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 windowId property (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.

Improved image export (4.5)

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.

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* [**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:

statusIconMenuPick *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:

statusIconClick *button*

statusIconDoubleClick *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 10to 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.

Specific bug fixes (6.0.1)

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

10828 Intersect function ignores threshold parameter.
10835 RevVideoGrabber, revSpeech and revBrowser missing from community builds (dues to licensing issues).
10836 Put doesn't output to message box if messages are locked.
10837 NumberFormat not working.
10843 Gif images do not repeat when repeat count is set to -1.

10844 Intersect does not recognise literal number as valid threshold value.
10848 Encrypting standalone stacks disables libURL.
10853 Crash when pasting image from Excel.
10858 Setting angle of scaled image to 0 reverts scaling.
10864 Crash when processing an image with an invalid source.

Specific bug fixes (6.0)

Crash on fetching certain forms of data from ODBC databases.
Crash in certain circumstances when referencing deleted objects.
Crash on Windows when windowshape is smaller than stack.
Char-level metadata imports incorrectly via rtfText
LinkText with link textStyle not distinguished from linkText without link textStyle.
Hyperlinks import incorrectly via rtfText.
Crash on linux when using export/import snapshot at size.
'Expanded' / 'condensed' never appear in htmlText.
Crash on exit from engine in some cases on Windows.
Setting font props on a closed stack doesn't update substacks.
2629 The long name does not always return.
3158 The colors of an image effect the image data.
9918 CantDelete property has no effect.
9942 Short time format not supported on Windows 7 and later.
9945 AM/PM not localised correctly on Windows XP.
9947 Verbatim chars not copied into dateFormat string.
9981 Crash when setting a watch on a global variable.
9988 Filename to pdf printer can't contain accented chars on Mac.
10061 Read from UDP socket crashes.
10062 Socket commands do not reset the result.
10096 Ask file not working properly on paths containing duplicate '/' on Windows.
10445 Sort international does not sort German text.
10478 Dispatch command does not work when lock messages it true.
10504 Control left/right arrow does not work properly with Russian text
10509 RevBrowserSnapshot does not work on Mac.
10516 UPD broadcast does not work on Mac.
10542 Rich text pasted from Firefox loses formatting.
10556 Metafile frame not taken into account when pasting from clipboard.
10558 Formatting changed in backgroundField upon saving.
10574 Printing to PDF does not include card images.
10577 Crash when using paragraph metadata.
10578 Ctrl-Left Arrow jumps passed spaces in fields.
10592 Different results returned by styledText of a line when using selectedLines as target.
10614 Zero width tabs not preserved in html text and styled text.
10620 Controls are not pasted correctly in edit group mode.
10627 Result is not cleared by 'the cipherNames' if there is no SSL.
10638 Crash in certain cases with drag-drop on Mac.
10644 Field ... of the owner of me doesn't evaluate correctly.
10649 Launch document/url blocks in some cases on Linux.
10652 Line height not taken into account when computing update region with fixedLineHeight.
10662 VTAB (linebreak) causes extra character to appear at end of line on some platforms.

10678 Error on socket causes unstable behavior on XP.
10683 Setting 'the colors' no longer unsets patterns.
10685 Paragraph properties now copy with sections of a paragraph (onto clipboard / through styledText/htmlText).
10693 Crash on Linux when opening script editor.
10695 Setting icon to name in behavior does not work as expected.
10698 Crash when setting textFont of stack that is in memory but not open.
10703 Import snapshot from object no longer works.
10703 Import snapshot from image causes crash.
10705 Visual effect in rect doesn't work as expected.
10705 Visual effect in rect does not work as expected.
10709 Core image visual effects crash.
10714 PNG images do not display in Windows XP.
10723 Setting text of image doesn't causes corrupt image when changing format.
10731 Saving certain stacks causes crash.
10741 Crash switching cards.
10748 Visual effect glitches when switching cards.
10759 Importing snapshot from rotated image adds artefacts.
10762 The text of a selectedLine returns empty.
10764 Crash when saving specific stack.
10772 Launching app from command line effects behavior of shell command.
10781 libpng warnings from commandline
10801 Crash when saving stack.
10802 Can't build standalones from Linux.
10811 Certain stacks saved with 6.0-rc4 and 6.0-rc5 crash earlier versions of LiveCode.
10812 Image antialiasing routines incorrect.

IDE Changes

Introduction

The primary focus of the LiveCode 6.0 release began as the IDE but has now changed with the success of RunRev ltd's Kickstarter campaign. The focus is now on getting LiveCode 6.0 to a point of release worthy stability to coincide with the first release of source code into the open source domain.

The message box update and new application browser remain in this release with custom controls and the property inspector pushed back to a later release.

Ask password update (6.0 DP5)

The behavior of **ask password** has been updated in 6.0 DP5 such that all passwords returned are un-hashed. This behavior mirrors the current **ask password clear** and brings the desktop platforms into line with the mobile platforms.

The current **ask password** behavior hashes all passwords using an undocumented algorithm (mcEncrypt). The use of the hash here has been deemed unnecessary: If the programmer wants to hash the passwords returned , they can do that directly using an algorithm of their choice.

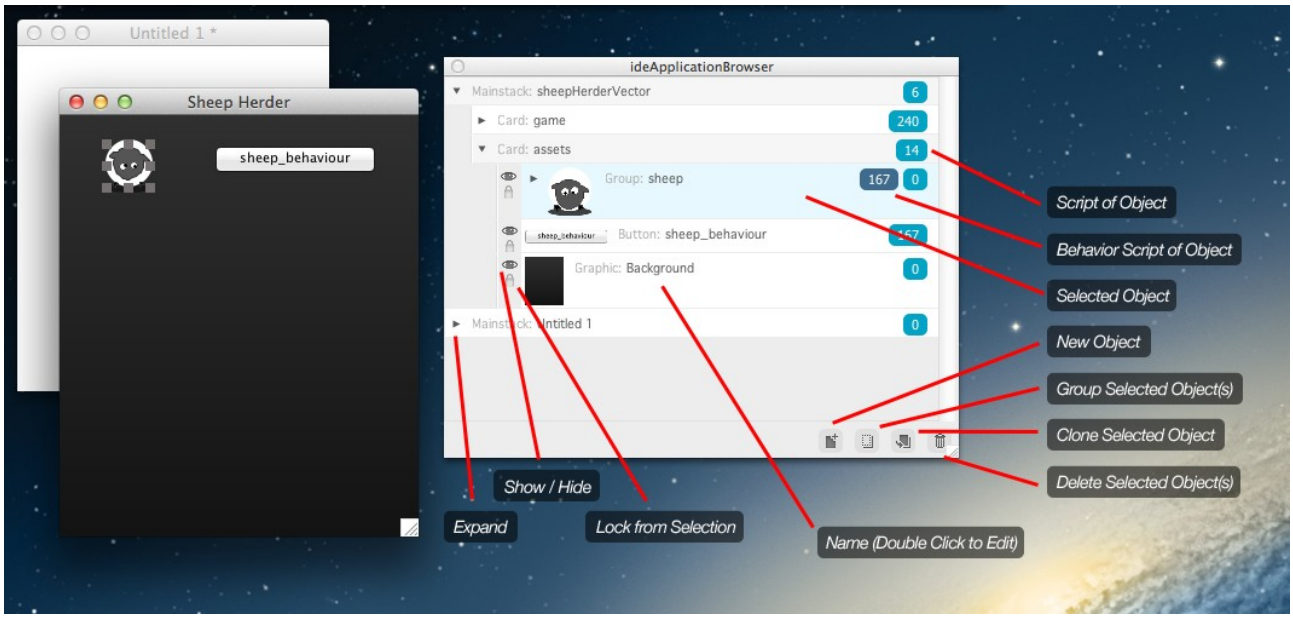
The old behavior can be reproduced using the following script:

```
ask password "Enter password"
get mcEncrypt(it)
```

See bug 10037 for further discussion.

Application Browser (6.0 DP1 - Experimental)

A new application browser has been added for LiveCode 6.0. The browser provides an additional view of the working stack in tree form.



Script of Object

Shows the number of lines of the script of the object.

Click – Opens the script in the script editor

Behavior Script of Object

Shows the number of lines of the behavior script of the object if one is set.

Click – Opens the behavior script of the object

Selected Object

The selected object is highlighted with a change in background color

Click – Deselects currently selected object(s) and selects the target object

Click + Drag – Relayers the object to the drop location

Click + Drag to actions – Performs action on dragged items (ie, new, group, clone, delete)

Shift + Click – Selects all the objects between the currently selected object and the target object

Cmd / Ctrl + Click – Selects the target object

Action Bar

The action bar contains a series of buttons that can be clicked on and dragged onto in order to perform the action of the highlighted or dragged object(s).

New – Create a new object of the same type as the highlighted object

Group – Group the object(s)

Clone – Clone the object(s)

Delete – Delete the object(s)

Name

The type and name of the object are shown for each object.

Double Click – Makes the name editable

Lock from Selection

Prevents / enables the selection of the object by the mouse pointer in the canvas area.

Click – Toggles the setting

Show / Hide

Shows / hides the object.

Click – Toggles the setting

Expand / Collapse

Expands and collapses the stack, sub-stack, card or group showing / hiding the contents

Click – Toggles the setting

Filtering (DP5)

You can now filter the controls shown in the Project Browser.

There are a number of way you can do this:

Just typing the the filter field will filter the controls by name.

Alternatively you can specify what you want to filter by, one of name, type, script lines. Name and type use "contains" to match the filter string.

name: <filter string>
e.g. name: back

type: <filter string> will filter by type
e.g. type: button

script [<|>|=] <integer>
e.g. script > 1

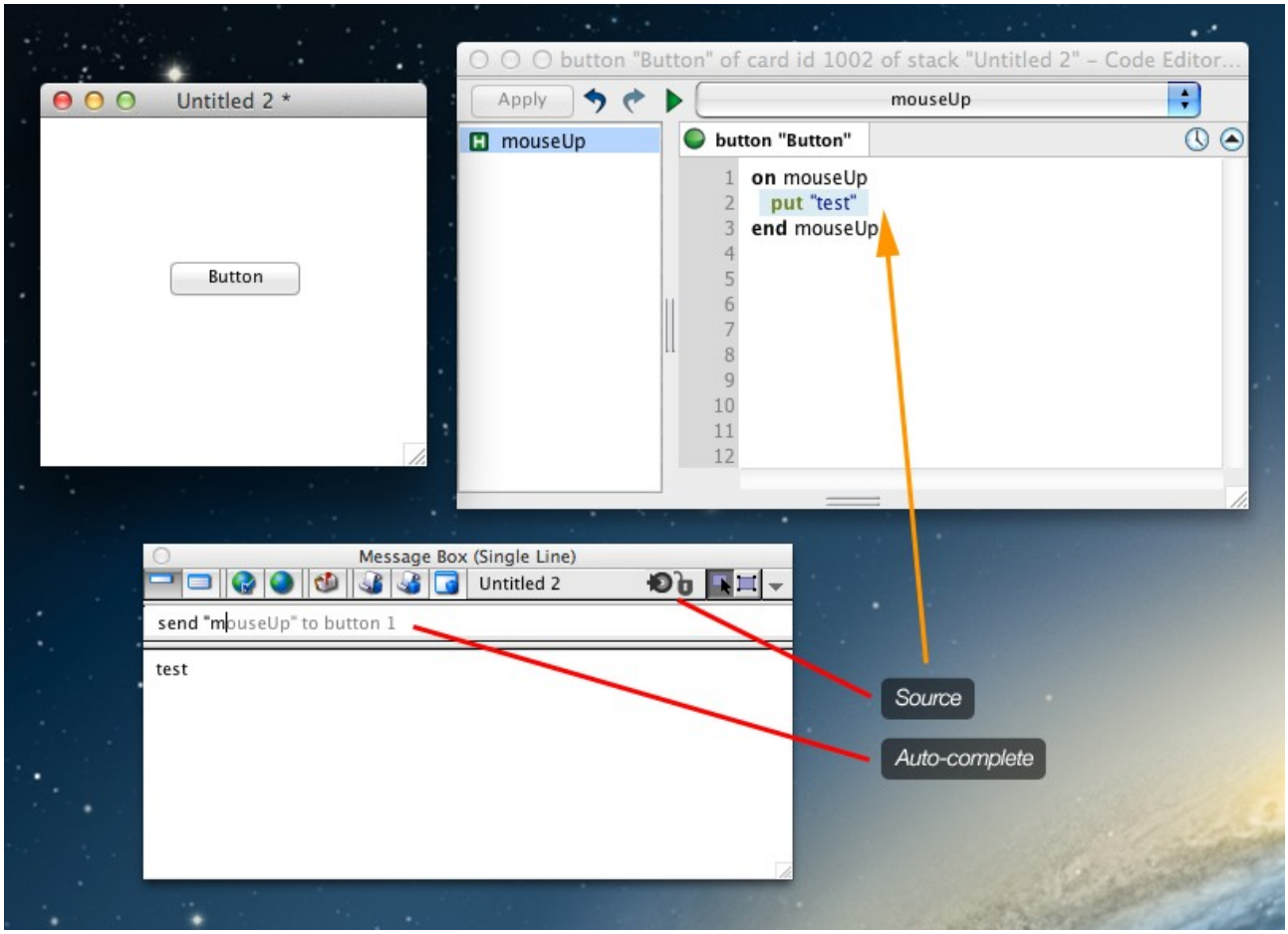
The matching controls are shown, along with the groups, cards, and stacks they belong to.

Settings (RC2)

A new setting icon has been added to the project browser allowing easy access to the preferences.

Message Box (6.0 DP1)

The message box has been update with two new features for LiveCode 6.0.



Source

The put command in LiveCode outputs content to the message box. The *source* button loads the script editor and selects the line of code that output the content

Click – Opens script editor

Auto-complete

An auto-complete feature has been added allowing the developer quick access to previous lines of code executed in the message box. Start typing to start the auto-complete feature.

Type – Starts auto-complete

Right Arrow Key – Accept suggested auto-complete

Up Arrow Key – Look forwards through auto-complete suggestions

Down Arrow Key – Loop backwards through auto-complete suggestions

The properties set on the stack determine exactly how the control is treated as well as providing the

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.

Datagrid

The datagrid is currently at version 1.0.2 (build 15).

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.
1.0.2 (build 14)	SortDataByKey specifically uses the first line of data in the column. Previously if there were multiple lines in a value it would cause problems.
	Fixed some issues in the auto scroll code used in drag/drop operations.

When setting the "scrollbar width" property the new value was not being passed into the function that sets the scrollbar width (bug 9684).

1.0.2 (build 15) Scrollbar width now reads correctly from registry.

Fixed bug with setting the dgText of a datagrid on another card (9575).

Fixed bug in datagrid setup (9576).

Specific bug fixes (6.0.1)

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

10830 Project browser group thumbnail does not update when controls moved.
10833 Community edition of LiveCode has “Commercial Edition” displayed in about dialog.
10841 Project browser shows contents of groups whose selectGroupedControls property is false.
10847 Hiding controls in the message box prevents command output.

Specific bug fixes (6.0)

IDE selecting datagrid does not present new inspector with datagrid properties
the clipboardData["image"] returns weird image data when copying a snapshot to the clipboard in Adobe Reader X (Windows 7)
new application browser window not visible
Printing to PDF does not include card images
Crash seemingly due to paragraph metadata
Incorrect word navigation when using OPTION + Left Arrow in script editor
IDE application browser not able to use password protected stack
10009 HTTP basic authentication fails with long passwords.
10037 Ask password hashes passwords.
10522 IDE Inspector does not present alignment pane when selecting more than one object
10531 Show behavior object in application browser
10532 Expanding disclosure triangle selects object
10533 Dragging button to other card in ideProjectBrowser causes error
10534 Control preview in new application browser not updated when objects added to control
10570 Click on Script Indicator to bring up Property Editor
10628 Can't paste control into a group.
10700 New property inspector incorrectly displayed in menu bar.
10701 Project browser tooltips are incorrect.
10702 Group disclosure triangle in project browser positioned incorrectly.
10707 Scrolling in project browser erratic.
10710 Menubar and toolbar positioning anomalies.
10711 Disclosure triangles in projects browser cannot all be collapsed at once.
10715 Tooltips in project browser incorrect.
10716 Alignment options in project browser cannot be undone.
10717 Selections in project browser not mirrored on stack.
10718 Project browser title differs depending upon platform.
10719 Project browser contents not updated when a stack is removed from memory.
10720 Text size options for script editor not comprehensive.

10721 Selecting preferences from script editor opens incorrect preferences pane.
10728 Expanding groups in project browser does not update scrollbar.
10729 View menu inconsistent.
10733 Project browser alignment needs improved.
10735 Empty script display needs improved in project browser.
10745 Selecting a control does not update the project browsers scrollbar.
10749 Project browser scrollbar not long enough.
10750 Project browser collapses group view immediately after expanding.
10751 Project browser preferences icon has incorrect tooltip.
10752 Project browser outputs debugging text to message box.
10754 Project browser scrollbar missing.
10756 Problem creating standalone in 6.0
10756 Error when creating standalones from stacks with cantDelete set to true.
10760 Only certain areas of the project browser scroll.
10761 Dragging controls in project browser awkward.
10778 IDE stacks show invisible objects when show invisible objects is turned on.
10787 Project browser thumbnails of fields broken
10791 Selecting any item on card "main" of "home" stack in new project browser crashes LiveCode
10792 revEnvironment...() functions gone?
10799 shutdown message not sent when home button is pressed on iOS when deployed from LiveCode 6 RC1-3
10804 Opening script editor doesn't work for LiveCode stacks.
10806 Error when right-clicking on non-hilited stack row in project browser.
10808 Project browser does not live resize.
10809 Refreshing the project browser resets its vScroll.
10819 Problem with stack already open in memory (multiple scenarios) in Linux
10824 Pblog.txt file created on desktop.
10825 Results field does not resize when the message box is resized.

User Guide

No changes.

Dictionary

Dictionary entries have been created and updated for the appropriate new and updated features.

Mobile Examples

No changes.

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
4.6.0 Release Notes http://www.runrev.com/downloads/livecode/4_6_0/LiveCodeNotes-4_6_0.pdf
4.6.1 Release Notes http://www.runrev.com/downloads/livecode/4_6_1/LiveCodeNotes-4_6_1.pdf
4.6.2 Release Notes http://www.runrev.com/downloads/livecode/4_6_2/LiveCodeNotes-4_6_2.pdf
4.6.3 Release Notes http://www.runrev.com/downloads/livecode/4_6_3/LiveCodeNotes-4_6_3.pdf
4.6.4 Release Notes http://www.runrev.com/downloads/livecode/4_6_4/LiveCodeNotes-4_6_4.pdf
5.0.0 Release Notes http://www.runrev.com/downloads/livecode/5_0_0/LiveCodeNotes-5_0_0.pdf
5.0.1 Release Notes http://www.runrev.com/downloads/livecode/5_0_1/LiveCodeNotes-5_0_1.pdf
5.0.2 Release Notes http://www.runrev.com/downloads/livecode/5_0_2/LiveCodeNotes-5_0_2.pdf
5.5.1 Release Notes http://www.runrev.com/downloads/livecode/5_5_1/LiveCodeNotes-5_5_1.pdf
5.5.2 Release Notes http://www.runrev.com/downloads/livecode/5_5_2/LiveCodeNotes-5_5_2.pdf
5.5.3 Release Notes http://www.runrev.com/downloads/livecode/5_5_3/LiveCodeNotes-5_5_3.pdf
5.5.4 Release Notes http://www.runrev.com/downloads/livecode/5_5_4/LiveCodeNotes-5_5_4.pdf
6.0.0 Release Notes http://www.runrev.com/downloads/livecode/6_0_0/LiveCodeNotes-6_0_0.pdf

Revisions

Revision 1	MM	Document created for 6.0-dp-1.
Revision 2	MM	Updated engine bug fix list for 6.0-dp-2. Updated IDE bug fix list for 6.0-dp2. Added section “Field Updates”. Updated section “ID Caching”. Added section “New global property – allowDatagramBroadcasts”. Updated section “IDE Roadmap”.
Revision 3	MM	Added section “Update to is and = operators”. Updated engine bug fix list for 6.0-dp-3.
	BB	Added section “IDE Color Profiles (6.0 DP3 – Experimental)”.
Revision 4	MM	Updated engine bug fix list for 6.0-dp-4 Updated section “IDE Roadmap”.
Revision 5	MM	Added section “Image handling changes” Updated engine bug fix list for 6.0-dp-5. Updated IDE bug fix list for 6.0-dp-5. Added section “Ask password update”. Updated section “Project Browser”.

		Removed section “IDE Roadmap”.
		Removed section “Color profiles”.
		Removed section “Property Inspector”.
		Removed section “Custom Controls”.
Revision 6	MM	Updated engine bug fix list for 6.0-rc-1.
		Updated IDE bug fix list for 6.0-rc-1.
Revision 7	MM	Updated engine bug fix list for 6.0-rc-2.
		Updated IDE bug fix list for 6.0-rc-2.
		Updated section “Project Browser”.
Revision 8	MM	Updated engine bug fix list for 6.0-rc-3.
		Updated IDE bug fix list for 6.0-rc-3.
Revision 9	BB	Updated engine bug fix list for 6.0-rc-4
		Updated IDE bug fix list for 6.0-rc4
		Added section “Script Security changes (6.0 RC4)”
Revision 10	MM	Updated engine bug fix list for 6.0-rc-5.
		Updated IDE bug fix list for 6.0-rc5.
Revision 11	MM	Updated engine bug fix list for 6.0-rc-6.
		Updated IDE bug fix list for 6.0-rc-6.
Revision 12	BB	Updated IDE bug fix list for 6.0-rc-7.
Revision 13	MM	Updated IDE bug fix list for 6.0-gm-1.
Revision 14	MM	Updated IDE bug fix list for 6.0.1-rc-1.
		Updated engine bug fix list for 6.0.1-rc-1.