LiveCode Builder Style Guide

Introduction

This document describes best practices to be followed when working with LiveCode Builder source code. Please follow it *especially* when writing code to be included with the main LiveCode product package.

Copyright headers

Please include a license header at the top of the `.lcb` file.

For the main LiveCode repository, or for any community extensions, the license is the GNU General Public License v3 *without* the "any later version" clause.

Naming

Module name

The module name uses reverse DNS notation. For example, a module created by the Example Organisation would use module names beginning with `org.example`.

Replace any hyphen (`-`) characters in a domain name with underscore (`_`) characters. For example, a module derived from the `fizz-buzz.example.org` domain could be `org.example.fizz_buzz`.

Additionally, add an underscore (`_`) to the start of any element in the domain name starting with a digit. For example, a module derived from the `999.example.org` domain could be `org.example._999`.

You must only use module names corresponding to domain names that you control or are allowed to use. This restriction is enforced by the the LiveCode extension store.

If you don't have a domain name of your own, you may use module names beginning with `community.livecode.<username>`, replacing `<username>` with the username you use to log into the LiveCode extension store. For example, if your username is "sophie", then you can create a module named `community.livecode.sophie.mymodule`.

For the main LiveCode repository, please use module names beginning with `com.livecode`

Always write module names in lower case.

Naming variables and parameters

Give variables and parameters `xCamelCaseNames` with a leading lowercase character indicating their scope and availability.

The meanings of the leading lowercase characters are:

Prefix	Context	Meaning
k	all	constant
s	module	static variable
m	widget	static variable
p	handler definitions	`in` argument
r	handler definitions	`out` argument
x	handler definitions	`inout` argument
t	handler bodies	local variable

In general, please use nouns to name your variables and parameters. Make the names descriptive; for example:

```
variable tOutputPath as String  -- Good
variable tString as String      -- Bad
```

For `Boolean` variables, please try to use "yes or no" names. For example:

```
variable tIsVisible as Boolean
variable tHasContents as Boolean
```

Naming handlers

Give handlers `TitleCase` names.

In general, please use verbs to name your handlers. For example,

```
handler RotateShape(inout xShape, in pAngleInDegrees)
    -- ...
end handler
```

Documenting the source code

You should add in-line documentation to `syntax` and `handler` definitions in LiveCode Builder source code. It is particularly important to add in-line documentation to all syntax and to any public handlers that aren't primarily accessed using syntax.

In-line documentation for a handler or syntax definition is extracted from a `/* */` comment block immediately before the start of the definition.

Please refer to the Extending LiveCode guide for full details of the syntax of in-line documentation comments, including examples.

Named constants

Often, it is useful to use constant values in your code. Please declare named constants rather than placing the values in-line. For example, you may want to create three tabs labelled "Things", "Stuff", and "Misc":

```
constant kTabNames is ["Things", "Stuff", "Misc"]

handler CreateTabs()
    variable tName
    repeat for each element tName in kTabNames
        -- Create the tab
    end repeat
end handler
```

In particular, please avoid any "magic numbers" in your code.

Whitespace

Indentation

Please indent with tab characters. Please use one tab character per level of indentation.

Please do not use a level of indentation at `module` level.

Comments should be indented to the same level as the code they apply to.

For example:

```
module org.example.indent

-- Example handler
handler Fizzbuzz(in pIsFizz)
    if pIsFizz then
        return "Fizz"
    else
        -- Maybe this should have a capital letter
        return "buzz"
    end if
end handler

end module
```

If it's necessary to mix spaces and tabs for indentation, please use 4 spaces per tab.

Handler declarations, definitions and calls

In handler type declarations and definitions, don't insert whitespace between the handler name and the parameter list. For example:

```
handler type Fizz()   -- Good
handler type Buzz ()  -- Bad
```

In handler parameter lists, please add a space between each parameter. For example:

```
handler FormatAsString(in pValue, out rFormat) -- Good
handler IsEqualTo(in pLeft,in pRight)          -- Bad
```

Please observe the same formatting in handler calls. For example:

```
variable tFormatted
variable tIsEqual
FormatAsString(3.1415, tFormatted)             -- Good
IsEqualTo (tFormatted,"3.1415") into tIsEqual  -- Bad
```