KVS Random Notes

This file contains *random* notes about KVS that I am gathering over time. It will be in the format of:

Question?
Answer.

I chose this format because I usually found myself asking “How can I …?” and then search to find the answer. This is to document the answers as clearly as possible and hopefully to build an organized index of the resulting data.

All information in this document is only from personal experience and should only be considered as a guide. The answers might not always be the best way to do things! It’s up to you to decide or work out what to do with your own code.

How can I send a private message to a channel on a specific IRC context (aka connection)?

Summary: Use the *rebind* command. http://www.kvirc.net/doc/cmd_rebind.html

Before you start, understand: An *IRC context* is how we identify which IRC connection to interact with in KVS. Since we can make KVIrc connect to multiple IRC networks at the same time, each connection is known as an IRC context.

The rebind command requires that you give it a *window ID* that is attached to an IRC context.
You can obtain a window’s ID using the *$window* function.
You can obtain a channel’s windows ID using the *$channel* function. (You can’t find channels using the $window function it seems)
The *$window.context* function will return which IRC context the window is attached to.

How can I make clickable text in KVIRC?

Description of clickable text: Clickable text in KVIRC operates like hyperlinks in webpages. We can define a portion of text as being clickable (and also define what type of clicking activates the action code) and assign some code to execute when it is clicked by the user. This feature allows us to easily let the user interact with our script at a basic level.

Before you start, understand: Escape Sequences. “Escaping” lets us ‘jump out of’ some piece of code that is being executed to do something else, and then we can ‘jump back in’ to the code we jumped out of and finish it. To make clickable text, we use the $cr function as the escape character.

Creating clickable text: The basic layout to make some clickable text is:
echo [prepending string]$cr![!event]<action code>$cr<clickable text>$cr[appending string]

The first thing to look at is all the coloured text above. Notice the blue text. These are the escape characters. They define different areas in our message that contains clickable text. Before you get confused about what an “escape character” is, just think of when you want to display a variable’s value in some output string. How do you do that? Like this:
echo [prepending string]%variableName [appending string]

When we display a variable’s value inline during an echo statement, we are effectively using a form of escape character (at least for the sake of learning what escape characters are). The % character tells KVS to “stop echoing the text as a string because we are going to now do something different”. That “something different” is to get a variable’s value and echo it. The variableName tells KVS which variable to get the value of.

The clickable text code is the same, but we also give KVS an *action code* which will be ran when the user sends the correct event. Look at the clickable text code again:

echo [prepending string]$cr![!event]<action code>$cr<clickable text>$cr[appending string]

The first $cr jumps us out of the echo command because “we are going to do something different”. Also, immediately after the first $cr is the *command rule* (![!event]), defining which event the user has to perform to make the <action code> run. We can use events like double clicking, primary clicking, opposite clicking etc.
After the event is defined, we then define the action code. An example will be shown below.
After the action code is defined another $cr defines we are starting the <clickable text> portion of the code. This is very similar to a hyperlink anchor tag in HTML which looks like:
Link
Where Link is the clickable text, and href=”url” is similar to our action code. It’s what happens when the Link is clicked.
After the clickable text another $cr “jumps us back in” to the original echo command and we can echo more text. The <clickable text> portion is echoed as part of the original echo command.

Example

echo Hello. Why don’t you try$cr![!rbt]echo You clicked it!$cr right clicking this$cr\?

Paste into script tester and execute. Roll your mouse over the words “right clicking this” and it will appear as a hyperlink. Right clicking it will execute the action code: echo You clicked it!

Additional:

- *Can I execute more than one command in the action code?*
  Not that I can work out. The only way to execute a block of code is to make an alias and call the alias. Note that you do not use "$alias", just "alias" to call it.

How can I set a tooltip for <this type> of widget?

All widget classes inherit from the *widget* class type. The tooltip setting function is in the *widget* class, so all other GUI widgets inherit the tooltip functions from there. See the widget documentation for details on the tooltip function and other available functions for all GUI widgets.

How can I safely find the user's home directory?

Use the $file.homedir function.

What are the numbers for the built in KVIrc icons when a widget (for instance, a toolbar button) wants an image assigned to it?

The numbers for inbuilt icons are in the KVIrc source. Look here:
https://github.com/kvirc/KVIrc/blob/198b8164f2da999bacd5f947ee565587d9a2bfe7/src/kvirc/kernel/KviIconManager.cpp#L84-L420

How should I structure my project for redistribution?

Note: This answer is not yet completed. It is only my opinion and how I have done mine. It could be foolish and wrong.

Use a single KVS file to house all the handlers, aliases, toolbar, actions etc you need. If your script doesn't use GUI objects, that's all you need probably.

Using GUI objects makes it a little more complex if you want maximum speed. KVIrc can store all your GUI objects in memory at all times even when the GUI isn't on screen. You can use an OnKVIrcStartup event handler to load your GUI elements to memory at startup. Then you can use an action (which can be linked to various elements like toolbers etc. in KVIrc) to just display your GUI program. So:
OnKVIrcStartup event handler will /parse a file that loads the GUI program to memory and then,
An action can be used to call the widgetObject->show() function to display the GUI on screen.

How can I check if a widget object is already being shown on screen so that another copy of it doesn't get displayed?

You can use *$objects.exists* function to see if the object variable exists in the first place, then you can use *%object->$isVisible* to check if it is already shown on the screen.

Why am I getting a complaint about an object not being valid when I use a reference to 'this' ($this, $$ or @)?

One thing I found was that if you were in a *callback* code block for something like the *dialog.** commands the scope inside the callback seems to be that of its own.
By this, I mean that the callback code block does not reside inside your object's code, therefore references to the 'this' object fails.

In addition, many times when I receive this error it will be when I am doing something like:
`if $objects.exists(%G_parent->%childObject) {...}`
I found that the error was usually complaining that `%G_parent` didn't exist, even though I wanted to check if its `%childObject` existed.

The error call is correct though, because if `%G_parent` doesn't exist, then `%childObject` can't exist.

To create error handling, you must first check that the parent exists, or make sure it definitely will exist at that point in code if you want to only check for existence of the child object.

How do I remove an item from a hash?

I found that you cannot use *unset* to remove an item from a hash. KVS throws errors about the value (hash) not being a list.

I couldn't find any other functions or commands that remove items from hashes so the method I used was to iterate through the hash values, if the value we find is the one we want to remove then do nothing (or other actions based on that condition), if the value you find is not the one to be removed add it to a temporary hash.

After the foreach loop of the original hash is done, make the original hash equal to the temporary hash.

How can I read from a channel's log file?

Below is an example code snippet you can use to get started:

```
# this code will find a channel's log file for you and read lines
from it.
# this is just so you can do something more with these basics.

# note: in the first line below "1" is the irc context for the
channel #moviegods, his can vary and you should find a way to obtain
```

```
the irc context value reliably.
%filename = $log.file($channel("#moviegods", 1))
echo -d filename = %filename
echo -d size = $file.size(%filename)
%lines = $file.readLines(%filename, ,20)

for (%i = 0; %i < %lines[]#; %i++)
     echo -d %i: %lines[%i]
```

I am using a SQLite database and it works on Linux but on Windows I can't insert data into the database?

If you are keeping an "age limited" set of data in your database table, you might be doing something like this in pseudocode:

```
Collect data for records
Build SQL query to insert data to database
Execute INSERT query
Build SQL query to delete database records that are beyond an age
limit
Execute DELETE query
```

If you are doing something like this then the problem will be with your timestamps, or your SQL which determines how old the record data is.

Check your timestamps first as this is the most likely problem. If you are using KVS function `$hptimestamp()` *it doesn't operate the same on Windows as it does on Linux*. On Windows it will return a value that reads as somewhere back in 1970 if it is converted into a date value. It probably starts from when KVIrc is launched, whereas on Linux it will return a value that is the same as what is usually called `NOW()` giving you the exact time and date in seconds since the Epoch.

To get correct timestamps under Windows use KVS function `$unixtime`.

When I use the exec command in KVS stdout is not showing me the output from the command I am calling? (under Windows)

The only way I could get exec to work properly and show me some stdout messages was to first make a batch file (under Windows) with the command line to call and then call the batch file. If you do this you will be able to catch the stdout output from your called program.