Luadch Manual

by pulsar
July 8, 2015

(based on the manual by blastbeat on October 18, 2008)

CONTENTS

1 INSTALLATION

1.1 Hardware Requirements

The hardware requirements will depend on the usage of the hub; high user counts will require more CPU, RAM and bandwidth. For 100 users, the following should suce:

- 128 MB RAM
- 500 MHz CPU
- broadband connection with 128 kbit/s upstream

1.2 Software Requirements

Luadch was tested on:

Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows Server 2003, Windows Server 2008, Windows Server 2012, Debian, Ubuntu, FreeBSD, OpenBSD, Linux Mint, CentOS, Red Hat, Busybox (Synology, QNAP, ASUS O!Play), Raspbian (Raspberry Pi B, B+, v2)

Linux / Unix:

- *Package:* `openssl`
  - to run encrypted with '`adcs://`'
  - to generate certificates

Windows:

- Windows builds including OpenSSL libs to run encrypted with '`adcs://`', no further installation of OpenSSL is required.

1.3 Windows Installation

1.3.1 Binary

Download Luadch binaries from GitHub:

http://luadch.github.io/

Unzip the archive somewhere, no further installation is needed.

1.3.2 Compiling

Install:

- OpenSSL with PATH environment
- MinGW
  - GCC
  - G++

Download Source:

https://github.com/luadch/luadch

Compile Source with:

```
compile_with_mingw.bat
```

The compiled hub can be found in the folder '`build_mingw`'.

1.4 Linux / Unix Installation

1.4.1 Binary

Download Luadch binaries from Sourceforge:

http://luadch.github.io/

Unzip the archive somewhere, no further installation is needed.

1.4.2 Compiling

Install Package:

- `openssl-dev` or `libssl-dev`
- `gcc`

Download Source:

https://github.com/luadch/luadch

Compile Source with:

```
compile_with_gcc.sh
```

The compiled hub can be found in the folder '`build_gcc`'.

2 CONFIGURATION

2.1 Getting started

First of all, to run an ADC hub, you will need some basic knowledge about ADC. You should know what `SID`, `PID`, `CID` and `TIGR` means. Learn more about ADC here:

http://adc.sourceforge.net/ADC.html

You don't have to read or understand the whole thing, but you should know the differences of the

addressed issues. The next step would be some knowledge about the programming language Lua. It isn't necessary but very helpful, because Luadch is written in Lua.
Some useful links for Lua:

http://www.lua.org
http://lua-users.org

2.2 Directory Overview

- `cfg/`
  - cfg.tbl - contains the configuration
  - user.tbl – user database
- `core/`
  - contains the basic hub logic
- `scripts/`
  - contains optional user scripts
- `scripts/lang/`
  - contains optional lang files for the scripts
- `log/`
  - contains log files
- `lib/`
  - contains library files
- `certs/`
  - contains your certificates
- `lang/`
  - contains language files for the hub core

2.3 UTF8 Notes

All user files, scripts and settings in Luadch must be UTF8 encoded. So make sure to save your files in UTF8 mode, because at the moment Luadch will only complain about non UTF8 strings, but not convert them.

2.4 The cfg.tbl

In '`cfg/cfg.tbl`' you can find general settings and all script permissions for Luadch. The file is in fact a Lua script, which returns a Lua table (therefore '`.tbl`'). The settings can be changed according to their description. Luadch will check the file for syntax/type errors, but nothing more.

2.5 The user.tbl

In '`cfg/user.tbl`' you can find the user database of your hub. Similar to '`cfg.tbl`', it's a Lua script, which returns a Lua array. You should not edit that file manually.

2.6 Advanced configuration

2.6.1 Using ADCS

First of all you need some general information about SSL:

http://en.wikipedia.org/wiki/Transport_Layer_Security
http://www.openssl.org/docs/

To use SSL in hub - client connections, you have to do 2 things:

- create certs
- activate SSL in cfg.tbl

A example conguration for certicates can be found in 'certs/'. Use make 'cert.sh/bat' to create some 'pem' files. In 'cfg.tbl', the key 'use_ssl' has to be 'true', and 'ssl_ports' needs an entry.

2.6.2 Using Scripts

Luadch provides some prewritten scripts. In fact virtually all useful facilities of Luadch are scripts in the 'scripts/' directory. Without them, simple things won't work. That means also, when you miss a feature, you have to script it. Every script you want to use must be added to 'scripts' table in 'cfg.tbl'. A script usually starts with a short description and a settings part. Make sure to save everything in UTF8.

The first thing you have to think about are user levels. The default levels are:

```
levels = { -- your levels with level names (array of strings)

        [ 0 ] = "UNREG",
        [ 10 ] = "GUEST",
        [ 20 ] = "REG",
        [ 30 ] = "VIP",
        [ 40 ] = "SVIP",
        [ 50 ] = "SERVER",
        [ 55 ] = "SBOT",
        [ 60 ] = "OPERATOR",
        [ 70 ] = "SUPERVISOR",
        [ 80 ] = "ADMIN",
        [ 100 ] = "HUBOWNER",
}
```

You can found this table in the 'cfg.tbl'. You may also use 1000 levels, but you should choose them carefully. To change them later can be very painful; you have to match it with all of your script permissions.

2.6.3 Language Files

The 'lang/' and 'scripts/lang/' directories contains the language files of hub and scripts. At the moment only English and German is available as default. To change the language of the whole hub, use key 'language' in 'cfg.tbl'. Default is 'en' for English. You can also change manually to 'de' for German.

3 RUNNING LUADCH

3.1 General Notes / HowTo

To run a Luadch Hub:

1. Without encryption, start the Hub and login with:
    - Nick: `dummy`
    - Password: `test`
    - Address: `adc://127.0.0.1:5000`
2. With encryption:
    1. Go to: '`certs/`' and start '`make_cert.sh`' on Linux/Unix or '`make_cert.bat`' on Windows to generate the certificates.
    2. Alternatively you can use the '`Luadch Certmanager`'.
    3. After that you can login with:
        - Nick: `dummy`
        - Password: `test`
        - Address: `adcs://127.0.0.1:5001`
3. Register an own nickname for you, there are two possibilities to do that:
    - Use rightclick menu: `User/Control/Reg`
    - Use command: `+reg nick <Nick> <Level>`

      Where `<Nick>` is your new nickname and `<Level>` should be the highest level `100`.
4. Now delete the dummy account, there are two possibilities to do that:
    - Use rightclick menu: `User/Control/Delreg`
    - Use command: `+delreg nick <Nick>`
5. After this first test you should adapt the hub to your needs:
    - Open: '`cfg/cfg.tbl`' with a UTF8 compatible Texteditor best with Lua syntax highlighting.
    - Read the descriptions and set the values to your need, Luadch uses a fair and reasonable default user permissions, but nevertheless you should read all.
6. If it's done, start your hub again and login, if it still runs there are to possibilities to enable your changes in the hub:
    - Use rightclick menu: `Hub/Core/Hub reload`
    - Use command: `+reload`
7. If you want to set other styles for lines or something:
    1. Go to: '`scripts/lang/`' here you can find all language files for each script.
    2. After that: `+reload`

3.2 Hub Commands

Type '`+help`' in the main chat to get all available commands according to your level. Or use the rightclick menu.

4 SCRIPTING

Most of all important things about the scripting API can be found in 'docs/Luadch_Lua_API.txt'. To avoid repeating myself I only explain a bit in the following sections.

4.1 Conventions

We tried to introduce a certain scripting style in Luadch and the prewritten scripts for better readability and less errors. In general we handle core scripts more strict.

Some conventions of 'core/':

- No global vars.
- Every function/module has to be imported with 'use' (to avoid globals).
- Declaration of all locals with script wide scope at the top.

Some conventions of 'scripts/':

- No global vars
- Name of the script should be as following: type 'name.lua'; for example 'cmd_reg.lua'
- A script should have at most one hub command. Instead of writing a script with commands '+reload' or '+reldel' use parameters like '+release add', '+release del' and name the script 'cmd_release.lua'
- Usage of language files which should stored in 'lang/' folder.
- Usage of script modules, for example 'ucmd', 'hubcmd', 'report', 'help'.

You don't have to follow these suggestions, but they made life easier for us.

4.2 Data Types

Luadch has some custom data types, which are of course more symbolic and not really comparable with the build in data types of Lua. Some functions will do type checks during runtime, but don't count on it. You have to take care to pass the right data to the functions. You can find a list of all data types in 'docs/Luadch_Lua_API.txt'.

4.3 Listeners

Scripts are usually listening for certain events. For these events they have to set listeners, which will be executed when the events happen. The general syntax to set a listener looks like this:

```
hub.setlistener( event, key, func )
```

This means set the function 'func' with identier 'key' (which can be a table or an unique string) as listener to event 'event'. Of course every event passes other arguments in the listeners functions. An example for listing on main chat messages would be:

```
hub.setlistener( "onBroadcast", { },
       function( user, adccmd, msg )
              hub.debug( msg ) -- prints the message in hub cli window
       end
)
```

You can find a list of all events in 'docs/Luadch_Lua_API.txt'

4.4 Script Modules

A nice feature in Luadch is the ability to export scripts as modules. So different scripts can use each other. One script needs to return something, and then another can import it. There are some prewritten script modules:

- cmd help.lua
- etc report.lua
- etc hubcommands.lua
- etc usercommands.lua

You can only import a module 'onStart', that means you have to set a listener for that event:

```
local help -- no globals...
hub.setlistener( "onStart", { },
	function( )
		help = hub.import( "cmd_help.lua" )
		assert( help )
		do_something_with( help )
	end
)
```

5 LINKS

Luadch Project:

http://luadch.github.io

DC-Planet (inofficial support forum):

http://www.dc-planet.de

Lua – The programming language:

http://www.lua.org

GitHub – Project hosting:

https://github.com

Sourceforge – Project hosting:

https://sourceforge.net