Libki Manual

Kyle M Hall

Version , 2018-07-19

Table of Contents

License

1. Introduction

1.1. What is Libki?

Libki is a Free Open Source cross-platform pc reservation booking and time management system designed to allow time limited access to computers on a network. Libki is ideally suited for use in locations where a controlled computing environment is paramount such as public access systems, libraries, school computer laboratories and more! It consists of two parts, the Libki server, and the Libki client.

Learn more about Libki by visiting the official Libki website, libki.org.

1.1.1. The Libki server

The Libki server utilizes a web-based administration system that can be accessed from any networked pc via a web browser. From the web administration one can create and delete users, change the amount of time left on an account, send or leave a message for a user, log out and/or ban users, and see which client computers are available and which are currently being used. In addition, the system has a public web interface for viewing which kiosks are available, which are unavailable, and to allow the placing of reservations.

1.1.2. The Libki client

The Libki client is cross-platform and runs on many operating systems including Microsoft Windows and Linux. The Libki client features a counter to let the user know how many minutes are left, and the ability to log off early. Any unused minutes can be used at any other computer running the Libki client. The Libki client can be set to log off from the operating system, or even reboot the computer when a user logs off from the Libki client.

1.2. Libki Server recommendations

At the time of this writing, it is recommended to run Libki Server on Debian Linux, Ubuntu Server or via Docker.

Guides for deployment are available in the Installation section.

Libki Server is meant to work on all web browsers, but is most tested on Chrome and Firefox.

1.3. Making contributions

This manual is an ever-changing document and edits to the manual are welcome at any time. The canonical source for the Libki manual is https://github.com/Libki/libki-manual

If you have a suggestions or contribution for the manual, please follow the guidelines for Contributing to the Manual.

2. Installation

2.1. Automatic Installation

When using the automatic installer, the server must be Ubuntu Server 18.04 or Debian Stretch.

Please read through the installation moments. There is a one-line command in the end if that's what you prefer.

2.1.1. Login as root

If you are not logged in as root, switch to root.

```
sudo su
```

2.1.2. Updating, upgrading and installing git

It's always best to do a fresh upgrade to be sure you have the latest versions of software available.

Git is used to download the Libki server.

```
apt update && apt upgrade -y && apt install git -y
```

2.1.3. Downloading and running the installer

First download the Libki server.

```
git clone https://github.com/libki/libki-server.git
```

Enter the downloaded directory.

```
cd libki-server
```

Run the installer.

```
./install.sh
```

2.1.4. One-line installer

This is the whole installation process in one single line.

```
apt update && apt upgrade -y && apt install git && git clone
https://github.com/libki/libki-server.git && cd libki-server && ./install.sh
```

2.2. Manual installation

If you wish to completely build the Libki server by yourself, these guidelines should help you along the way. This is written towards someone new to terminal and a lack of GUI.

First of all, update and upgrade.

```
apt update && apt upgrade -y
```

2.2.1. Dependencies

```
apt install cpanm curl perl git make build-essential unzip mysql-server ntp -y
```

You might need these packages too, depending on your server. If you encounter error messages when installing the needed perl modules, this is the place to start.

```
apt install libmysqlclient-dev libxml-parser-perl libxml-libxml-perl
```

2.2.2. Setting up a user

It is suggested to setup a new user account to run the server from. From here on, we'll use the username **libki**. Give it a good, strong (and preferrably memorable) password.

```
adduser libki
```

2.2.3. Clone the repository

Clone the repo to the home directory of your newly created user.

```
git clone https://github.com/libki/libki-server.git /home/libki/libki-server
```

2.2.4. Install needed perl modules

```
cd /home/libki/libki-server
cpanm -n --installdeps .
```

This will take a while. Make sure it ends with saying everything was installed correctly. If not, you will need to fix what's missing.

Now we need to add the Libki server's perl module to our $PATH, so our shell knows where to find it.

```
echo "export PERL5LIB=$PERL5LIB:/home/libki/libki-server/lib" >> ~/.bashrc
```

We also need to add it to the libki user's $PATH.

```
echo "export PERL5LIB=$PERL5LIB:/home/libki/libki-server/lib" >> /home/libki/.bashrc
```

2.2.5. Create a database

Depending on your MySQL or MariaDB version, it may or may not want you to log in with a username and password. If you are to log in with a username and password, the username is **root** and you chose the password during the dependencies installation.

If that's the case, start MySQL with `mysql -uroot -p`.

If you didn't get to create a root password during installation, just start MySQL with `mysql`.

Once inside MySQL, we need to create a database and a user. Note that all aphostrophes are essential and cannot be omitted.

```
CREATE DATABASE libki;
CREATE USER 'libki'@'localhost' IDENTIFIED BY 'CHOOSEAPASSWORD';
GRANT ALL PRIVILEGES ON libki.* TO 'libki'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

2.2.6. Populate the database and create an admin account

We got our perl modules, we have an empty database… Let's fill it with things.

```
./installer/update_db.pl
```

In order for Libki to access the database, we must give it the login information. Make a copy of the libki_local.conf.example file and remove .example. Open it and change the password to the one you chose.

```
cp libki_local.conf.example libki_local.conf
nano libki_local.conf
```

You save and close with Ctrl-O and Ctrl-X, respectively.

Now create an admin account, so we can access the administration pages once the server is up and running.

```
./script/administration/create_user.pl -u ADMINUSERNAME -p ADMINPASSWORD -s
```

2.2.7. Setting up log files

The Libki server will produce log files. Their default location is /var/log/libki. Even if you don't want to change the default location, you still need to make a working copy the log4perl.conf.example file.

```
cp log4perl.conf.example log4perl.conf
```

2.2.8. Installing cron jobs

Part of what makes the Libki server tick is scheduled jobs called cron jobs. `./script/cronjobs/libki.pl` is the timer and `./script/cronjobs/libki_nightly.pl` is the cleaner that resets everything overnight.

There are two pre-written cron files, just to import. The first one is for the libki user and the second one for root.

```
cat installer/cron/libkicron | crontab -u libki -
cat installer/cron/rootcron | crontab -
```

2.2.9. Create a Libki service

Copy the init template to /etc/init.d.

```
cp init-script-template /etc/init.d/libki
```

If you want to edit the port of the server (if you, for example, want to run it on port 80 and don't want to use a reverse proxy), this is the time. Open it up, change port number from 3000 to 80 (or something else), save and close.

Finally, run update-rc.d to enable Libki as a service.

```
update-rc.d libki defaults
```

2.2.10. Start the server

```
service libki start
```

If all went well, you should have a server up and running by now. You can visit it on http://127.0.0.1:3000/administration.

2.2.11. Manual install optional: Set up your reverse proxy

Make sure you're logged in as root.

- Install Apache

```
apt-get install apache2
```

- Navigate to the libki-server directory

```
cd /home/libki/libki-server
```

- Run the apache_setup.sh script

This disables the old default conf, copies reverse_proxy.config to Apache's folder and enables both the Libki reverse proxy and the needed modules..

```
./script/setup/apache_setup.sh
```

- Restart apache

```
service apache2 restart
```

2.3. OPTIONAL: Configuring Libki to authenticate against a SIP server

To enable SIP authentication, you will need to edit your libki_local.conf and add a section like this:

```
<SIP>
    enable 1
    host ils.mylibrary.org
    port 6001
    location LIB
    username libki_sipuser
    password PassW0rd
    terminator CR
    require_sip_auth 0
    enable_split_messages 0
    fee_limit 5.00 ①
    deny_on charge_privileges_denied ②
    deny_on recall_privileges_denied ③
    deny_on excessive_outstanding_fines ④
    deny_on_field AB:This is the reason we are denying you ⑤
</SIP>
```

① Can be either a fee amount, or a SIP2 field that defines the fee limit (e.g. CC), delete for no fee limit

② You can set SIP2 patron status flags which will deny patrons the ability to log in

③ You can set as many or as few as you want. Delete these if you don't want to deny patrons.

④ The full listing is defined in the SIP2 protocol specification

⑤ You can require arbitrary SIP fields to have a value of Y for patrons to be allowed to log in. The format of the setting is *<Field>:<Message>*.

The SIP section requires the following parameters:

- enable: Set to 1 to enable SIP auth, 0 to disable it.
- host: The SIP server's IP or FQDN.
- port: The port that SIP server listens on.
- location: The SIP location code that matches the sip login.
- username: The username for the SIP account to use for transactions.
- password: The password for the SIP accouant to use for transactions.
- terminator: This is either CR or CRLF depending on the SIP server. Default is CR
- require_sip_auth: Does this SIP server require a message 93 login before it can be used? If so this should be set to 1 and the username/password fields should be populated. This should be set to 1 for Koha.
- enable_split_message: IF thie server supports split messages you can enable this. This should be

set to 0 for Koha.

- fee_limit: As notated, this can be a set number or a SIP field to check. If the fee limit is exceeded, the user login will be denied.
- deny_on: This can be repeated to deny logins based on the patron information flags detailed in the SIP2 protocol specification.
- deny_on_field: This can be repeated to deny logins if the Specified field does not have a value of "Y".

2.3.1. Troubleshooting

You can now test to see if your server is running by using the cli web browser 'links'. If you don't have links installed you can installed it via the command

```
sudo apt-get install links
```

Now, open the Libki public page via:

```
links 127.0.0.1:80
```

If this loads the Libki login page, congrats! If you get an error, you can try bypassing the proxy and access the server directly on port 3000.

```
links 127.0.0.1:3000
```

If this works, then you'll want to check your Apache error logs for the failure reason. If it does not work, you'll want to check the Libki server error log instead. It can be found at /home/libki/libki_server.log if you've followed this tutorial closely.

3. Configuration

3.1. Settings

3.1.1. Basics

Default time allowance per day

This setting determines the total number of minutes a user will have for the day by default.

The value of **Default time allowance per day** should be a whole number.

Default time allowance per session

This setting determines the total number of minutes a user will have for each *session* per day. So, if

a system is configured with 120 minutes time allowance per day, and 30 minutes time allowance per sessions, a user will be able to log in 4 times per day for 30 minutes at a time.

The value of **Default time allowance per session** should be a whole number.

Default guest time allowance per session

This is the same as **Default time allowance per session**, except for guest accounts instead of regular uses.

The value of **Default guest time allowance per session** should be a whole number.

Client registration update delay limit

When a Libki client launches, it periodically connects to the Libki server to register itself. If a client does not re-register itself within this number of minutes, the Libki server will remove it from the list of active clients.

The value of **Client registration update delay limit** should be a whole number.

3.1.2. Data retention

History retention

This setting controls how long user and client data is retained for the purpose of generating statistics and checking usage history, in days. If this setting is left empty, the data will be kept indefinitely.

The value of **History retention** should a whole number.

Inactive user retention

This setting controls how long a user can be inactive before being automatically deleted, in days. If this setting is left empty, all users will be kept indefinitely.

The value of **Inactive user retention** should be a whole number.

This setting should be used to conform to GDPR if necessary.

3.1.3. Client behavior

This setting controls how clients may be used by users. There are three possibilities.

- First come, first served.
  - Allows a patron to log in to any client not currently being used.
- Reservation only.

- Requires a patron to place a reservation for a client before logging in to it.

- First come first served, with option to place reservation.
  - A hybrid approach that allows patrons to log in to any client that is not currently in use or reserved.

The setting Display usernames controls if the user's username is displayed on the reserved client computer.

3.1.4. Reservations

Reservation timeout

The amount of time \(in minutes \) that a user has to log into his or her reserved computer. If the user does not log in within the specified time limit, the reserved client will become available again.

The value of **Reservation timeout** should be a whole number.

Display usernames

This setting determines if a Libki client that is reserved and waiting will display the username of the person it is waiting for.

3.1.5. Automatic time extension

Libki has an automatic time extension system that allows users to remain logged into a client computer beyond the pre-determined number of minutes per session a user is allotted.

Extension length

The amount of time to automatically increase a user's session time by \(in minutes \).

The value of **Extension length** should be a whole number.

Extend time at

This setting controls at what point in time an extension length is triggered. A session will be extended when the user's session time drops below this number, in minutes.

The value of **Extend time at** should be a whole number.

Extend time unless

This setting determines if a user is eligible for an automatic time extension. It has two options:

- User's client is reserved
  - This choice prevents a time extension in the case that the user's client is reserved. Reservations for other client's are not taken into account.

- Any client is reserved
    - This choice prevents a time extension in the case that **any** client is currently reserved. If any client is reserved a time extension will not take place, even if the users's client is not currently reserved,

When extending time

This setting determines how minutes are added to a patron's account when an automatic time extension occurs. It has two options:

- Take minutes from daily allotment
    - This options moves minutes from the user's daily allotment of minutes to the user's session minutes. That means the user can continue using the client computer, but only up to his or her daily allotment of time.
- Don't take minutes from daily allotment
    - This option adds minutes to a users session "out of thin air". As such, it does not effect how many sessions a user will have per day.

Client login banner settings

The client banners are optional areas on the top and bottom of the Libki client login screen. They are functionally like to web browsers. As such, anything that is viewable in a web browser is viewable in the banner areas \(size permitting \).

Source URL

The URL for the image or html that you wish to display in the banner section.

Width

If the **Source URL** is an image, it can be forced to a specific width instead of using the image's actual width. Leave empty to use the image's actual width.

Height

This is the same as **Width** for the **Source URL** but for height.

3.1.6. Guest passes

Prefix for guest passes

The phrase that each guest pass username should start with. If left empty, the phrase "guest" will be used (e.g. guest1, guest2, guest3, etc).

This setting can be a word or short phrase, but should contain only letters and numbers. Avoid using spaces or special characters.

Passes to create per batch

If the *Multiple guests* button is used, this setting will control how many guest accounts are generated with each clock.

The value of **Passes to create per batch** should be a whole number.

Username label

The text in this field will be prepended to the guest username, (e.g. *"Username:"*).

Password label

This setting works the same as **Username label** but for the generated password instead of the username.

3.1.7. Third party integration

URL

Entering a url here will cause the username in the user's table of the web administration to become a hyperlink with the user's username at the end. For example, *http://catalog.koha.library/cgi-bin/koha/members/member.pl?quicksearch=1&searchmember=* will link to the Koha ILS's search function for the given username.

Make sure the URL beings with http:// or https:// as necessary.

3.2. Closing Hours

Closing hours are a way to prevent users from starting a session that will be cut short by the closing of the location he or she is at. Closing hours can be set on a site-wide basis, or on a per-location basis. If a given location has no closing hours set, that location will use the *All locations* closing hours.

3.3. Single Sign-on

3.3.1. SIP2

Single Sign-on can with an ILS can be achieved via SIP2. Settings for the ILS SIP2 server can be stored in the *libki_local.conf* file or the **SIP configuration** setting.

3.3.2. LDAP

Single Sign-on with other systems can be achieved via LDAP. Settings for LDAP server are currently stored in the *libki_local.conf* only, though setting support is expected soon.

3.4. Print Management

Print management in Libki is powered by Google Cloud Print. To set up print management, first set up your printers in Google Cloud Print. Next, generate a client id and secret. Finally, enter your configuration in the **Printer configuration** setting as YAML. The code block below is an example configuration with two printer profiles for a single printer (one color, one monochrome).

```
google_cloud_print:
  client_id: 893746288161-libc4aj9loitf5i2lcuuonj6ggqb37uc.apps.googleusercontent.com
  client_secret: dEjNmggj-PS9_LnvP92jIYu3

printers:
  color:
    type: google_cloud_print
    google_cloud_id: d4355eb9-5b5b-3982-1492-9a1245298409
    name: color
    ticket:
      color:
        type: STANDARD_MONOCHROME

  monochrome:
    type: google_cloud_print
    google_cloud_id: d4355eb9-5b5b-3982-1492-9a1245298409
    name: monochrome
    ticket:
      color:
        type: STANDARD_MONOCHROME
```

To get your client id and client secret: . Get oAuth2 Credentials .. Browse to https://console.developers.google.com/ .. Enable Cloud API .. Create credentials for OAuth 2 type="Other" .. Save the ID and secret for use in the Libki print configuration

To authorize your server to use the Cloud Print API: . Run `script/administration/enable_google_cloud_print.pl` . Open the URL given in a web browser . Authorize your account . Copy and paste the code you are given back into the script . Hit enter . You should recieve the message `Session stored.` if everything was successful.

To get your printer cloud id: . Set up the printer for Google Cloud Print . Browse to https://www.google.com/cloudprint/simulate.html . Use the Printer Search API to search for you printers .. Or browse to https://www.google.com/cloudprint/search directly . Locate your printer in the results, find the "id" field, it should look something like `id: "ed4ddb78-dc03-8574-8687-be3995df8cd4"`

4. Administration

4.1. Statistics & History

5. Desktop client

6. Backing up and restoring your Libki database

It is **HIGHLY** adviced to only doing backups and restorations of the database while the system is not being used!

6.1. Backing up

The backup program is intended to be used on a server with a libki user.

A backup folder will be created in the libki user's home directory, where the backups are stored.

Please note that this will only create local backups. It is adviced to transfer them to a safe location. A guide on how to transfer automatic backups to an FTP server will be added later.

6.1.1. Make a backup manually

Simply run `libki-backup`.

The backup will be created and stored in `/home/libki/backups`. The file name will contain date and time for the backup for easier identifying, i. e. `db_180706_19.30.sql`.

6.1.2. Make automatic backups

The backup program features a silent mode, perfect for when you don't want any user input or text output. Run `libki-backup -s` to run it in silent mode.

Automatic backups work the same way as the manual backups, but does not provide any output text. The backup file is created directly in `/home/libki/backups`.

If you want to make backups automatically according to a schedule, this is easiest done through crontab. I won't go through how crontab works, but will provide some example schedules.

Start by running `crontab -e`. You will probably want to use the default editor. Add the schedule at the bottom of the file.

Crontab examples

Daily backup at 02:00:

```
0 2 * * * /usr/local/bin/libki-backup -s
```

Backup every Sunday evening:

```
0 21 * * SUN /usr/local/bin/libki-backup -s
```

Backup the first date in every month at 01.15:

```
15 1 1 * * /usr/local/bin/libki-backup -s
```

6.2. Restoring a backup

Restoring a backup of the database will delete the current database, so be sure you really want to do this.

Start by navigating to the folder containing the backup you want to restore. If the backup was created with the backup program, this folder will be `/home/libki/backups`.

```
cd /home/libki/backups
```

Now run the program and specify what backup you want to restore.

```
libki-restore db_180706_19.30.sql
```

Your database is now at the point it was when the backup was created.

7. Contributing to Libki

7.1. Contributing to the Libki source code

7.1.1. Submitting bug fixes and features

Formatting Patches

- File an Issue on GitHub in the matching repository (client, server, manual, etc...) for the issue (for both enhancement **and** bug fixes)
  - The earlier you file the better. It's much better to file an issue stating your intention to develop a new feature or fix before you start to give other community members a chance to review the issue and give feedback.
- The subject line of your commit(s) should begin with "Issue XX: " followed by the description of what the patch does.
- Include a hyperlink to the GitHub issue this patch resolves.
  - GitHub will notice and automatically link the commit to the issue.

Libki server

Pull requests for features and fixes to the Libki server should be submitting as pull requests via the Libki server GitHub repository.

Libki client

Pull requests for features and fixes to the Libki client should be submitting as pull requests via the Libki client GitHub repository.

7.1.2. Coding Guidelines

Server coding guidelines

- Perltidy new or altered pieces of code before submission.
    - Libki server comes with a .perltidyrc file that should always be used when tidying Libki server code.

7.2. Contributing to the Manual

Suggestions for edits can be sent to the Libki Documentation Team as a merge request via GitHub.

To join the Libki Documentation Team, please contact Kyle Hall via the Libki developers mailing list or the Libki Slack workspace.

The manual is currently available on GitHub.

8. Credits

8.1. Authors

- Kyle M Hall <kyle@kylehall.info>
- Erik Öhrn <erik.ohrn@gmail.com>
- Christopher Vella <chris@calyx.net.au>
- Luke Fritz <ldfritz@gmail.com>