Libki Setup and Configuration Guide *for* SEO Member Libraries

updated May 2017

Jessica D. Dooley, Adams County Public Library
dooleyje@adamscolibrary.org
937-386-0089

Summary

Libki is free, open-source computer session management software for libraries, developed by Kyle Hall. This configuration guide is a supplement to Libki's installation documentation, designed to help SEO member libraries or others using SirsiDynix Symphony configure Libki for your library environment.

Table of Contents

Resource Guide

Developer and About

Libki is a free, open-source kiosk session management solution for libraries, developed by **Kyle Hall**. Learn more about Libki at http://libki.org. Contact Kyle Hall directly at kyle.m.hall@gmail.com.

Read a concise summary of Libki features here: http://bywatersolutions.com/what-is-libki/

In this 10-minute YouTube video, Kyle walks through the Libki product, including client, administration, and reservation interfaces. Keep in mind that this demo from 2013 does not include the most recent features, but gives a good overview and feel of using Libki.
http://bywatersolutions.com/2013/07/24/libki-intro/

Email Lists

Libki Users email list is active, and is the best first source for community support. Kyle also activity monitors and participates on the list. https://lists.sourceforge.net/lists/listinfo/libki-users

Libki Developers email list is also active for those contributing features and code to the Libki project. Libki is currently undergoing a massive collaborative rebuild to add print management features.
https://lists.sourceforge.net/lists/listinfo/libki-developers

Support

ByWater Solutions offers open source software support for libraries. They specialize in supporting Koha ILS. Paid support for Libki, including installation, configuration, and staff training, is available from ByWater Solutions. Libki developer Kyle Hall is a ByWater Solutions team member.
http://bywatersolutions.com

Kyle accepts commissions to develop custom features. Adams County Public Library commissioned three custom features, which were added to the Libki code in 2015.

Instructions and Code

Libki source code and installation instructions are hosted at GitHub. https://github.com/Libki

Libki client Windows executables are hosted at BitBucket. Other Libki code still available through BitBucket has been superseded by the GitHub code. https://bitbucket.org/account/user/libki-kms/projects/PROJ

The Standard Interchange Protocol v2 specification from 3M. View what database fields can be queried by default with SIP2, and how to take action on that field value. Ask SEO for SirsiDynix's annotated SIP2 extended spec, which includes information about SirsiDynix's custom SIP2 fields.
http://multimedia.3m.com/mws/media/355361O/sip2-protocol.pdf

SIP communication testing tool: this nifty little applet was developed by Mike Fields at the Central Library Consortium, and released for use by other libraries under the GPLv3 license. Send test SIP queries to your SIP2 server via GUI, and view the full output. http://www.clcohio.org/sip-testing-tool

Server Setup and Configuration

Libki server may be installed on Linux platforms. Instructions are optimized for **Debian**. Canonical instructions are kept up to date in the GitHub Wiki as the Libki server code is updated. Instructions and the latest version of the server are available at Github:

https://github.com/Libki/libki-server#manual-installation.

Before you begin, give attention to the following considerations.

Network considerations:

- Select a static IP for your Libki server.
- The Libki server communicates with clients via TCP traffic on port **3000**. A different port may be arbitrarily selected when first configuring the Libki server.
- Select a desired server IP and port before installing clients. The server's IP and port must be configured in the client software upon installation. Should the IP and port change, the client configuration file (.ini) must be edited to reflect the correct values on each client computer.
- Based on your network configuration, firewall or routing rules may be needed to permit the Libki server to communicate with intended endpoints. If desired, you can configure the Libki server's reservation webpage to be accessible from the public Internet. If you intend to offer Libki reservation over the public Internet, you may wish to use the optional configuration for a transparent proxy (provided in the GitHub manual installation instructions).

Essential server configuration:

- When configuring a new install of Debian or other Linux OS to host a Libki server, consider enabling NTP to ensure that Libki has the correct time when managing client sessions.
- The built-in daemon **`systemd-timesyncd`** can be configured to manage client-only network time synchronization, rather than installing a full NTP package.
- A Catalyst Session plugin that Libki uses to manage sessions creates large amounts of tmp data, and fails to clean old data, which can rapidly cause server performance degradation. Add the following **cron job** to the root user's crontab to remove tmp data each night:

  ```
  rm -rf /tmp/libki
  ```

  If tmp data grows to unmanageable size more often than daily, use the following cron job at a more frequent interval:

  ```
  rm -rf /tmp/libki/session/data/*
  ```

  Note that this will clear ALL session data. Current sessions should have an open file handle, and should not be deleted. Testing this in your environment before production is recommended.

Managing logs:

- Libki writes each client's registration every 60 seconds, and each user's session login and logout, and any SIP queries, to **`libki_server.log`**. This log will grow rapidly in size, and logs significant amounts of patron information associated with computer use. Consider using an automated process such as **logrotate** to manage the log.
- The built-in tool **logrotate** can be used in conjunction with **rsync** and **cron** to rotate, compress, and move logs to a specified location for storage. *Sample code:*
    - Configure logrotate for libki. (Customize for your environment and preference.)

```
nano /etc/logrotate.d/libki

/home/libki/libki_server.log {
      daily
      missingok
      rotate 7
      notifempty
      compress
      delaycompress
      create 664 libki root
      sharedscripts
      dateext
      dateformat -%Y-%m-%d
}
```

    - Write a simple **rsync** script to a desired destination. Save the script as a .sh.
    - Add a cron job to run the rsync script at desired intervals.
- Jay Miley (jay@mercerlibrary.org) wrote a Perl POSIX script to manage **`libki_server.log`**.
    - Install the POSIX Perl module.
    - Create a destination directory for the historical logs.
    - Write a script with the following contents:

```
use POSIX qw(strftime);
use File::Copy;
use DateTime;

my $now = time();
my $before = $now - (24*60*60);
my $yesterday = POSIX::strftime( '%Y-%m-%d', localtime($before)
);

move ("/home/libki/libki_server.log", "/home/libki/libki-
logs/$yesterday.log") || die("error");
open(FW,">/home/libki/libki_server.log");
```

SIP2 Configuration with SirsiDynix Symphony

Libki can be configured to use the Standard Interchange Protocol v2 to query a remote database for user information.

To enable SIP2 in your Libki server, modify the **`libki_local.conf`** file to enable the SIP2 parity bit (set to 1). You must also provide a SIP2 server IP address, port, location, username, and password. Contact SEO's IT department to request a SIP2 IP address, port, location, username, and password for your Libki instance.

Sample SIP configuration from **`libki_local.conf`**:

```
<SIP>

enable                  1               # 1 is on, 0 is off
host                    192.168.1.1     # use the IP address of SEO's SIP server
port                    6001            # use the port of SEO's SIP2 server
location                SIPNAME         # an arbitrary value; ask SEO
username                SIPUSER         # an arbitrary value; ask SEO
password                SIPPASS         # an arbitrary value; ask SEO
terminator              CR              # carriage return
require_sip_auth        0               # 1 is on, 0 is off
enable_split_messages   1               # 1 is on, 0 is off
# fee_limit             5.00
```

 # Uncomment the line to enable. Defines the billed amount on a patron record that will deny patrons computer access. Can be either a fee amount, or a SIP2 field that defines the fee limit (such as CC).

```
# deny_on               charge_privileges_denied
```

 # Uncomment the line to enable. Select the SIP2 patron status flag(s) which, if true for a patron account, will deny the patron computer access. All available patron status flags are defined in the SIP2 specification. To enable additional patron status flags, add another instance of deny_on on new line.

```
# deny_on_field         XY: Message.
```

 # Uncomment the line to enable. This flag allows you to check an arbitrary SIP2 field in the patron record, and deny if the field value is not Y. XY is the SIP2 field label, and Message is an arbitrary message you may configure to be returned to the patron to explain to explain the denied access. This feature can only be used with SIP2 fields where a possible value is Y. This feature was commissioned by ACPL for use with Symphony, to check the Internet Use field (PI) and deny if the field is not set to Y.

deny_on_field **PI: No Internet privileges. Please see staff for assistance.**

```
</SIP>
```

Sample responses to a SIP2 query from a Libki server to SEO's SirsiDynix Symphony database: (These queries and responses by your Libki server are logged in **`libki_server.log`**.)

Sample of SIP responses from a Libki log, with field label glossary and origin of field value:

```
'PH' => 'STANDARD',       # Notify by (User Cat 4) (Sirsi custom field)
'PG' => '3NA',            # School district (User Cat 3) (Sirsi custom field)
'BE' => 'test@test.com',  # Email address (SIP2 spec)
'CB' => '0999',           # Charged item limit (SIP2 spec)
'CQ' => 'Y',              # Valid patron password (aka PIN) (SIP2 spec)
'patron_status' => \{     # These can be invoked to deny access using SIP
configuration in libki_local.conf. Use the syntax deny_on desired_patron_status.
Repeat on a new line for each status desired to trigger denial.
                              'hold_privileges_denied' => ' ',
                              'recall_overdue' => ' ',
                              'too_many_renewals' => ' ',
                              'too_many_items_charged' => ' ',
                              'charge_privileges_denied' => ' ',
                              'too_many_items_lost' => ' ',
                              'card_reported_lost' => ' ',
                              'excessive_outstanding_fines' => ' ',
                              'too_many_claims_of_items_returned' => ' ',
                              'too_many_items_overdue' => ' ',
                              'excessive_outstanding_fees' => ' ',
                              'too_many_items_billed' => ' ',
                              'renewal_privileges_denied' => ' ',
                              'recall_privileges_denied' => ' '
                            \},
'PF' => '2OTHO',          # County Township aka User Cat 2 (Sirsi custom field)
'AA' => '29061000000000', # Patron identifier (User ID) (SIP2 spec)     \
'PA' => '20170509',       # Patron expiration date (Sirsi custom field)
'PD' => '19500101',       # Patron Birthdate (Sirsi custom field)
'DM' => '$500.00',        # Deposit maximum (Sirsi custom field)
'CA' => '0003',           # Overdue items limit (SIP2 spec)
'DB' => '$0.00',          # Deposit balance (Sirsi custom field)
'AF' => 'OK',             # Screen message (readable status) (SIP2 spec)
'BV' => '0.00',           # Fee amount (SIP2 spec)
'BH' => 'USD',            # Currency type (SIP2 spec)
'AQ' => 'ACP',            # Permanent location (Home Library) (SIP2 spec)
'PC' => 'ACA',            # Patron loan class (Profile Name) (Sirsi custom field)
'AE' => 'LASTNAME, FIRSTNAME A', # Personal name (SIP2 spec)
'PE' => 'FDOB50_59',      # Gender Decade of Birth (User Cat 1) (Sirsi custom field)
'PI' => 'Y',              # Internet use (User Cat 5) (Sirsi custom field)
'BL' => 'Y',              # Valid patron (SIP2 spec)
'BZ' => '0099',           # Hold items limit (SIP2 spec)
'hold_items_count' => '0000',    # Available holds; triggers a message to patron if
value is greater than 0; ACPL commissioned feature (SIP2 spec)
'BD' => 'SABINA OH 45169' # Home address (SIP2 spec)
```

Notes:

- These SIP queries and responses by your Libki server are logged in **`libki_server.log.`**
- The possible value or default value for any given SIP field above is dependent on SEO's configuration of our SirsiDynix Symphony database, on your library institution's profile with SEO, and on your preferred practices for entering information in patron accounts. For example, the value returned for 'CB' (charged item limit) is determined by your library's profile with SEO, and refers to the maximum simultaneous checkouts a patron registered by your library can have. The 'CB' value may vary by the current patron's 'PC' (profile), depending on your policy.
- If a field in a patron record returns a null value to the SIP query, the null response will not be logged. For example, if there is no email address in a patron's email address field, the SIP field 'BE' will not be logged for that SIP query.
- Notice that significant amounts of patron information are collected by this SIP transaction. Consider implementing an automated process to manage and discard the contents of **`libki_server.log`** over time.

Client Setup and Configuration

Libki clients are available for Windows as an .exe. Clients can be compiled from the source for Linux and Unix OS. https://github.com/Libki/libki-client/wiki/libki-client-installation
Download the most recent Windows Libki Client here: https://bitbucket.org/libki-kms/libki-client/downloads/

The Windows .exe Libki client is configured with a GUI wizard at setup. Configuration can be edited after installation at any time in the .ini file. The Windows Libki client .ini can be found in C:\ProgramData\Libki\Libki Kiosk Management System.ini .

A more detailed description of client configuration options is now available in the Libki Client Wiki on GitHub. https://github.com/Libki/libki-client/wiki/libki-client-installation

Sample .ini configuration:

```
[server]
host=192.168.1.10              # Libki server IP
port=3000                      # Libki server port
scheme=http                    # must be http
[node]
name=clientcomputername        # Identifies this client in the web interface
logoutAction=reboot            # select reboot, logout, no_action
nopasswords=0                  #  1 to hide, 0 to show the password field
password=MD5hashvalue          # Staff disable/unlock password as an MD5 hash value
onlyStopFor=clientusername     # Windows user for which Libki won't run on login
location=BranchOrArea          # Identifies client groups in the web interface
onlyRunFor=clientusername      # Windows user for which Libki will run on login
username=Username or library card number
[windows]
EnableStartButton=1            # Enable or disable Windows start button
[labels]
username="Library Card:"       # Configure login screen field labels
password="PIN:"                # Configure login screen field labels
```

The "password" is a value which, when entered into the password field of the Libki client, exits the Libki client and gives access to the desktop.

If you do not require your users to enter a password or PIN to log in to Libki, you may wish to hide the password field. However, consider that if the password field is not available, library staff cannot use the override password to unlock the Libki client and access the desktop.

To reset the staff override password in the Libki client at any point after initial client installation, you must generate an MD5 hash of the desired client unlock password, and configure it in the Libki client .ini. Use a free tool like this one to generate the hash value. http://www.md5hashgenerator.com/

Web Interface Setup and Configuration

Libki is entirely managed through a web interface. Access the administration web interface at http://192.168.1.10:3000/administration/login, using the static IP and port of your Libki server. A super administrator user can control all Libki settings. An administrator user can manage all Libki user and client operations, and view statistics and history, but not change settings or configuration. A super administrator account must be created with the **`create_user.sh`** script on the Libki server before the web interfaces will be accessible. Additional users can be created via the web interface, or the script.

Access the public reservation or client display web interface at http://192.168.1.10:3000, using the static IP of your Libki server. Make this web interface available on a kiosk in the library, or on the public Internet via your website, to allow patrons to view computer availability and place or cancel their own reservations for public computers.

The web interface settings permits customization of your instance of Libki with custom JavaScript. Separate JavaScript can be specified for the Administration interface vs. the Public interface.

Eric Lochtefeld (eric@mercerlibrary.org) wrote custom JavaScript to cause the Administration interface to load the Client tab by default, rather than the User tab:

```
$(document).ready(function() {
   $('#client-tab-label').click();
});
```

Libki can generate ephemeral guest passes in bulk, in batch sizes which you configure in settings. Guest passes can be written to a directory. They also display in a new browser tab immediately upon batch creation, by default. Jessica Dooley (dooleyje@adamscolibrary.org) wrote an AutoHotKey script to send the batch of guest passes to Symphony Receipt Printer, print, and cut individual guest pass slips. Sample code is available in the appendix. Please note that an AutoHotKey script would need to be significantly customized for your environment.

Libki permits you to configure an image to appear as a banner in the top panel and lower panel of client login screens. Separate images can be configured for the upper and lower banners. Images must be hosted at a URL. The precise size of images must be specified in the settings.

Libki can be configured to retain settings and history for a set period of time. Please note that configuring Libki not to store history or statistics does not prevent Libki from writing all user activity to **`libki_server.log`**. The log must be managed to prevent retaining this data indefinitely. Consider cleaning the log as a way to comply with your library's patron information privacy policies.

Closing hours can be configured in Libki so that Libki client sessions and reservation availability will respect library open hours. Hours can be configured for all clients, or on a per-location basis, with location being an arbitrary value set in the client configuration .ini per client. Differing hours for specified exception dates, such as holidays or closed days, can also be configured per location.

Troubleshooting

More specific troubleshooting information will be added shortly.

Appendix

Sample AutoHotKey script for printing and cutting batches of individual guest passes by invoking Symphony Receipt Printer's cut-paper text string. This script makes use of two static, pre-created text files saved on the workstation. The AutoHotKey is **ctrl + g** (for "guest"). This script will need significant customization, based on the printers installed on your workstation, and other unique behavior of your workstation. After building a successful .ahk script, use the .ahk to generate a small standalone .exe that will run the script when invoked by your chosen hotkey. Install the .exe in the Startup folder of the workstation that will be frequently creating and printing batches of guest passes, such as a reference or circulation workstation.

I use this script with Symphony Receipt Printer (Jay Miley, jay@mercerlibrary.org) and Epson TM-T88-line thermal receipt printers. When executed correctly, the script should take you from viewing a batch of ephemeral guest pass data, generated by Libki on demand and displayed in a browser, to a string of individually printed, cut, and properly formatted guest passes on receipt paper in less than 10 seconds.

To edit this script, and deploy it for your environment, use AutoHotKey: http://ahkscript.org/.

```
;
; AutoHotkey Version: 1.x
; Language:       English
; Platform:       Win9x/NT
; Author:         Jessica D. Dooley <dooleyje@adamscolibrary.org>
;
; Script Function:
;     Template script (you can customize this template by editing
"ShellNew\Template.ahk" in your Windows folder)
;

#NoEnv  ; Recommended for performance and compatibility with future
AutoHotkey releases.
SendMode Input  ; Recommended for new scripts due to its superior speed and
reliability.
SetWorkingDir %A_ScriptDir%  ; Ensures a consistent starting directory.

# Selected ctrl + g for "guest passes"
# ctrl + g should be invoked while viewing the ephemeral guest passes in the
browser, after generating them in Libki.
# The first section selects and copies the guest pass information from the
browser tab, inserts Symphony Receipt Printer's cut paper string between each
pass, opens a prepared text file, and pastes the guest pass info in it.
```

```
^g::
ClipBoard =
Send ^a
send ^c
ClipWait 2
StringReplace, clipboard, clipboard, Your, -->End of Slip<--`r`nYour, All
IfWinNotExist ahk_class Notepad
Run Notepad.exe "C:\Users\Public\libki.txt"
WinWait ahk_class Notepad
WinActivate ahk_class Notepad
WinWaitActive
Send ^v

# The next section opens a second prepared text file, which will handle
printing each guest pass as an individual print job. It also sets the system
default printer to Symphony Receipt printer before printing.

Send {NumpadPgUp 4}{Up 50}{Shift Down}{Down 7}{Shift Up}^x
ClipWait 2
Run Notepad.exe "C:\Users\Public\guestpass.txt"
IfWinExist, guestpass.txt - Notepad
WinActivate
WinWaitActive
Send ^v
Sleep 100
Run RUNDLL32.exe PRINTUI.DLL`,PrintUIEntry /y /n "Symphony Receipt Printer"
Sleep 1000
Send ^p
Sleep 300
Send {Tab 6}
Sleep 120
Send {Enter}
Sleep 120
Send ^a
Send {Del}
Sleep 120

# The next section selects one guest pass, copies and pastes the information
to the prepared text file, prints and cuts it, and clears that pass's data.

# Iterate the following 20 lines once for each guest pass, based on the
number of guest passes you have configured Libki to generate per batch.
```

```
IfWinExist, libki - Notepad
WinActivate
WinWaitActive
Send {NumpadPgUp 4}{Shift Down}{Down 7}{Shift Up}^x
ClipWait 2
IfWinExist, guestpass - Notepad
WinActivate
WinWaitActive
WinWaitActive
Send ^v
Sleep 100
Send ^p
Sleep 120
Send {Tab 6}
Sleep 120
Send {Enter}
Sleep 120
Send ^a
Send {Del}
Sleep 120

# Clean up and close the text files without saving. Return the system default
printer to the desired default, if other than Symphony Receipt Printer.

Send, {Alt Down}F{Alt Up}x
Sleep 60
Send, {Left}
Send, {Enter}
Sleep 120

Send, {Alt Down}F{Alt Up}x
Sleep 60
Send, {Left}
Send, {Enter}
Sleep 120

Run RUNDLL32.exe PRINTUI.DLL`,PrintUIEntry /y /n "SHARP MX-2700N PCL6"
Return
```