User Guide
JALoP over BEEP (v1.x)

Draft

07 October 2022

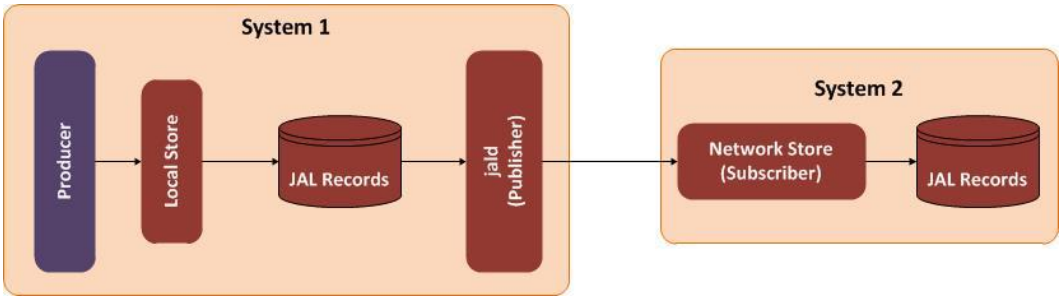
Table of Contents

1 JALoPv1.x 4
2 Download/Clone Source Code 4
2.1 GitHub 4
2.2 JALoP CTC-Internal Git Repositories 5
3 Build and Install C Publisher (jald) & Subscriber (jal_subscribe) 5
3.1 System Provisioning 5
3.1.1 REHL 7 / CentOS 7 5
3.1.2 RHEL 8 / CentOS 8 6
3.2 Build and Install the C Publisher & Subscriber 7
3.3 Configure the C Publisher 7
3.4 Configure the JAL-Local-Store 8
4 Build and Install BeepCore Java 9
4.1 System Provisioning 9
4.2 Clone “beepcore-java” repository from GitHub 9
4.3 Build and Install BeepCore Java 9
5 Build and Install the Java Subscriber (jnl_test-*.jar) 10
5.1 Clone “jjnl” repository from GitHub 10
5.2 Build and Install the Java Subscriber 10
5.3 Configure the Java Subscriber 10
6 Run Publisher and Subscriber to Transfer JAL Records 11
6.1 Disable SELinux or set to Permissive (For Testing Only) 11
6.2 Stop Firewall (For Testing Only) 11
6.3 Start JAL-LOCAL-STORE 12
6.4 Insert Records into JAL-Local-Store 12
6.4.1 Insert Log Records 12
6.4.2 Insert Audit Records 12
6.4.3 Insert Journal Records 12
6.5 Check for Inserted Records in JAL-Local-Store 12
6.6 Start C Publisher 13
6.7 Start Java Subscriber 13
6.8 Purge Records from Local Store 13
6.9 Run the C Publisher (jald) with GDB 14

6.10 Run the Publisher (jald) with Valgrind 14
7 JALoPv1.x Data-Taps 14
7.1 jalop-coreutils 14
7.1.1 Build & Install 14
7.1.2 Run tee and tail 14
7.2 jalop-jalauditd 15
7.2.1 Build & Install 15
7.2.2 Configure jalauditd 15
7.2.3 Run jalauditd 15
7.3 jalop-rsyslog 16
7.4 jalop-log4cxx 16

1 JALoPv1.x

JALoPv1.x is JALoP over BEEP. It has a C Publisher/Subscriber, a Java Publisher/Subscriber, and several data-taps. Note that the Publisher usually resides on a CDS system to securely and reliably transfer journal, audit, and log records generated by the CDS to one or more remote Subscribers.



In the diagram above, “System 1” depicts a JALoP C Publisher system, and “System 2” depicts a JALoP Subscriber system. The “System 1” above can also represent the JALoP subsystem of a CDS system where the “Producer” can be the CALD (Central Audit and Logging Daemon) or CJD (Central Journal Daemon) or both. Details can be found in “JALoP Software Design” documentation.

2 Download/Clone Source Code

Create a top-level ‘jalop’ directory to store all JALoP repositories. This will be referred to as `<jalop_root>` throughout this document.

```
% mkdir jalop
% cd jalop/
```

2.1 GitHub

JALoP Publisher and Subscriber repositories are available on GitHub at this URL – https://github.com/JALoP

Clone “JALoP” and “jjnl” repositories using git –

```
% git clone https://github.com/JALoP/JALoP.git
% cd JALoP/
% git checkout -t origin/1.x.x.x

% git clone https://github.com/JALoP/jjnl.git
% cd jjnl/
% git checkout -t origin/1.x.x.x
```

The “jjnl” repository requires Java BEEP Core, which is also available on GitHub.

```
% git clone https://github.com/JALoP/beepcore-java.git
```

2.2 JALoP CTC-Internal Git Repositories

All the JALoP source code repositories are available on the CTC-internal git server on the ISIS network. You must VPN into the ISIS network to access those repositories.

If you have been granted access to the internal git repositories as a developer, you can git clone the repositories as below –

```
% git clone git@jalop.cdsc.ctc.com:/data/git/jalop.git
% cd jalop/
% git checkout -t origin/1.x.x.x

% git clone git@jalop.cdsc.ctc.com:/data/git/jjnl.git
% cd jjnl/
% git checkout -t origin/1.x.x.x

% git clone git@jalop.cdsc.ctc.com:/data/git/beepcore.git
```

If you do not have access as a developer, you still can clone the CTC-internal repositories in read-only mode from within the ISIS network.

```
% git clone http://jalop.cdsc.ctc.com/gitreadonly/jalop.git
% cd jalop/
% git checkout -t origin/1.x.x.x

% git clone http://jalop.cdsc.ctc.com/gitreadonly/jjnl.git
% cd jjnl/
% git checkout -t origin/1.x.x.x

% git clone http://jalop.cdsc.ctc.com/gitreadonly/beepcore.git
```

3 Build and Install C Publisher (jald) & Subscriber (jal_subscribe)

The C Publisher (jald) is the process that negotiates with the Subscriber(s) and sends JAL records to them. This is in “jalop” (or “JALoP” if cloned from GitHub) repository. Clone the repository as mentioned above, if not done yet.

3.1 System Provisioning

3.1.1 REHL 7 / CentOS 7

The following instructions will allow a CentOS 7 minimal install to be provisioned to build and run the JALoP C implementation:

- Install EPEL
  - `$ sudo yum install epel-release`
- Install JALoP dependencies available from repos
  - `$ sudo yum install @development libxml2-devel libconfig-devel libuuid-devel openssl-devel libdb-devel xmlsec1-openssl-devel python2 python36-scons lcov libtool-ltdl-devel`

- Install bundled JALoP dependencies
  - Navigate into your cloned JALoP repo
  - Force-install axl RPM that has hard python 2.6 dependency
    - `$ sudo rpm -i --force --nodeps 3rd-party/axl/RHEL6/RPMS/*.x86_64.rpm`
  - Install nopoll and vortex rpms
    - `$ sudo yum install 3rd-party/{nopoll,vortex}/RPMS/RHEL6/*.x86_64.rpm`
  - Download and install test-dept unit test library.
    - `$ git clone https://github.com/norrby/test-dept.git`
    - `$ cd test-dept`
    - `$ ./boostrap`
    - `$ ./configure`
    - `$ sudo make install`

3.1.2 RHEL 8 / CentOS 8

The following instructions will allow a CentOS 8 Stream minimal install to be provisioned to build and run the JALoP C implementation:

- RHEL 8: Enable CodeReady Builder
  - `$ sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms`
- CentOS 8: Enable Powertools (unbranded version of RHEL's CodeReady Builder)
  - `$ sudo dnf config-manager --set-enabled powertools`
- Install JALoP dependencies available from repos
  - `$ sudo dnf install @development libxml2-devel libconfig-devel libuuid-devel openssl-devel libdb-devel xmlsec1-openssl-devel libtool-ltdl-devel apr-util-devel libcurl-devel doxygen lcov`
- `Build and install test-dept`
  - `$ git clone` https://github.com/norrby/test-dept.git
  - `$ cd test-dept`
  - `$ ./bootstrap`
  - `$ ./configure`
  - `$ sudo make install`
- `Build and install axl`
  - `$ git clone` https://github.com/ASPLes/libaxl.git
  - `$ cd libaxl`
  - `$ ./autogen.sh --disable-py-axl`
  - `$ sudo make install`
- `Build and install vortex`
  - `$ git clone` https://github.com/ASPLes/libvortex-1.1.git
  - `$ cd libvortex-1.1`
  - `$ ./autogen.sh --disable-{sasl,websocket,xml-rpc,tunnel,pull,http,alive}-support --disable-{py,lua}-vortex --disable-vortex-client`
  - `$ sudo make install`
- `Install python36-scons`
  - `$ sudo yum install python36-scons`
  - `$ sudo pip3 install scons`
- Use python36 as python
  - `$ sudo alternatives --config python`
    - Enter the number for the /usr/bin/python36 selection
- Ensure /usr/local/lib is in the dynamic linker path

- $ echo /usr/local/lib | sudo tee /etc/ld.so.conf.d/usr-local-lib.conf
- $ sudo ldconfig

3.2 Build and Install the C Publisher & Subscriber

The "jalop" (or "JALoP" if cloned from GitHub) repository builds the following applications –

- "jald" (the JALoP C Publisher)
- "jal-local-store" (the JAL-Local-Store)
- "jaldb_tail"
- "jal_dump"
- "jal_purge"
- "jalp_test" (a test Producer)
- "jal_subscribe" (the JALoP C Subscriber)

It also builds the following shared libraries –

- "libjal-common.so"
- "libjal-db.so"
- "libjal-network.so"
- "libjal-producer.so"
- "libjal-utils.so"

Follow the instructions below to build and install the above-mentioned components -

- Change to the jalop top directory –
  - `% cd <jalop_root>/jalop/  (or, cd <jalop_root>/JALoP/, if cloned from github).`
- Checkout the "1.x.x.x" branch of jalop (if not already checked out. Check "git branch" output) -
  - `% git checkout 1.x.x.x`
- Clean up and build jalop -
  - `% scons –c   # to clean up.`
  - `% scons`
- Install the publisher -
  - `% sudo ./install_rhel_x86_64.sh`

3.3 Configure the C Publisher

Make a copy of `<jalop_root>/jalop/test-input/jald.cfg` and adjust according to your Subscriber's IP, port, etc. This is to avoid changing the sample configuration file in the git source tree. Here is an example of `jald.cfg` file –

```
# The path to the private key, used for TLS.
private_key = "./test-input/cert_and_key";

# The path to the public cert, used for TLS.
public_cert = "./test-input/cert";

# The directory containing the certificates for the remote peers.
remote_cert_dir = "./test-input/certs";

# The path to the root of the database.
db_root = "/root/testdb";

# The path to a directory containing the JALoP schemas.
schemas_root = "./schemas";

# The port the Publisher will listen on.
port = 8444L;

# The IP address (interface) the Publisher will to listen on, or
0.0.0.0 to listen on all.
host = "127.0.0.1";

# For subscribe, the maximum number of records to send before sending
a 'digest' message
pending_digest_max = 10L;

# For subscribe, the maximum number of seconds to wait, before sending
a 'digest' message
pending_digest_timeout = 100L;

# How long to wait, in seconds, before polling for records after
finding no records
poll_time = 1L;

# List of allowed Subscriber peer configurations
peers = ( {
           hosts = ("127.0.0.1");
           subscribe_allow = ("journal", "audit", "log");
     } );
```

Check the section “Run the Publisher and Subscriber to Transfer JAL Records” below.

3.4 Configure the JAL-Local-Store

The JAL-Local-Store (`jal-local-store`) is a process that receives and stores JAL Data sent from the JAL Producer applications. It has a Berkeley Database (BDB) to store and process the JAL records. The `jal-local-store` process must be started before the Publisher process (`jald`).

Make a copy of `<jalop_root>/jalop/test-input/local_store.cfg` and update accordingly. This is to avoid changing the sample configuration file in the git source tree. Here is an example local store configuration file –

```
private_key_file = "./test-input/rsa_key";
public_cert_file = "./test-input/cert";
system_uuid = "34c90268-57ba-4d4c-a602-bdb30251ec77";
hostname = "test.jalop.com";
db_root = "/root/testdb";
schemas_root = "/home/marefin/jalop/jalop/schemas/";
socket = "/home/marefin/jalop/jalop/jal.sock";
sign_sys_meta = false;
manifest_sys_meta = false;
```

4 Build and Install BeepCore Java

4.1 System Provisioning

Install the following packages (if not done already) –

```
% sudo yum install java-1.8.0-openjdk-devel ant maven
```

4.2 Clone "beepcore-java" repository from GitHub

```
% cd <jalop_root>/
% git clone https://github.com/JALoP/beepcore-java.git
```

4.3 Build and Install BeepCore Java

Build –

```
% cd <jalop_root>/beepcore-java/
% ant dist-tgz
```

Install –

```
% build/beepcore-0.9.20/lib/install_jars.sh
```

This should install 3 JAR files to ~/.m2/repository/ as shown in the example output below –

```
[INFO] Installing /home/marefin/jalop/beepcore-java/beepcore.jar to
/home/marefin/.m2/repository/org/beepcore-
java/beepcore/0.9.20/beepcore-0.9.20.jar

[INFO] Installing /home/marefin/jalop/beepcore-java/beeptls-jsse.jar
to /home/marefin/.m2/repository/org/beepcore-java/beeptls-
jsse/0.9.20/beeptls-jsse-0.9.20.jar

[INFO] Installing /home/marefin/jalop/beepcore-java/concurrent.jar to
/home/marefin/.m2/repository/EDU/oswego/cs/dl/util/concurrent/1.3.4/co
ncurrent-1.3.4.jar
```

Verify that the 3 JAR files mentioned above are successfully installed; if for some reason those are not installed, manually install (copy) them to intended destination as shown above.

5 Build and Install the Java Subscriber (jnl_test-*.jar)

5.1 Clone "jjnl" repository from GitHub

This is the "jjnl" repository. Clone the repository as below (if not done already) –

```
% cd <jalop_root>/

% git clone https://github.com/JALoP/jjnl.git
% cd jjnl
% git checkout –t remotes/origin/1.x.x.x
```

5.2 Build and Install the Java Subscriber

- Build -
  - `% cd <jalop_root>/jjnl/jnl_parent/`
  - `% mvn clean`
  - `% mvn package`

- Install (optional) -
  - `% cd <jalop_root>/jjnl/jnl_lib/`
  - `% mvn install`

5.3 Configure the Java Subscriber

```
% cd <jalop_root>/jjnl/jnl_test/
```

Copy `./src/test/resources/sampleSubscriber.json` here and update accordingly. This is to avoid changing the sample config file in git source tree. If not running with TLS, comment out or remove the "ssl" section below.

An example of `sampleHttpSubscriber.json` given below –

```
{
  "address": "127.0.0.1",
  "port": 8444,
  "subscriber": {
      "sessionTimeout": "00:00:00",
      "dataClass": [ "audit", "log", "journal" ],
      "pendingDigestMax": 1,
      "pendingDigestTimeout": 120,
      "output": "./output",
      "mode": "archive",
  }
  "ssl": {
    "Key Algorithm": "SunX509",
    "Key Store Passphrase": "changeit",
    "Key Store Data Type": "file",
    "Key Store": "./certs/server.jks",

    "Trust Algorithm": "SunX509",
    "Trust Store Passphrase": "changeit",
    "Trust Store Data Type": "file",
    "Trust Store": "./certs/remotes.jks",
  }
}
```

Check the section “Run the Publisher and Subscriber to Transfer JAL Records” below.

6 Run Publisher and Subscriber to Transfer JAL Records

Follow the steps below to run the C Publisher and Java Subscriber to transfer JAL records.

6.1 Disable SELinux or set to Permissive (For Testing Only)

```
================================================
Permanent: have "SELINUX=disabled" or “SELINUX=permissive” in
"/etc/selinux/config" file, then restart VM.
Temporary: Enter the command "/usr/sbin/setenforce 0"
```

6.2 Stop Firewall (For Testing Only)

```
=========================================
RHEL/CentOS 6.x
        Stop:    % sudo service iptables stop
        Disable: % sudo chkconfig iptables off


RHEL/CentOS 7.x, 8.x
        Stop:    % sudo service firewalld stop
          Or,    % sudo systemctl stop firewalld
        Disable: % sudo systemctl disable firewalld
```

6.3 Start JAL-LOCAL-STORE

```
=======================================
# May need to clean up the first time. Make sure no jal-local-store is
running. For examples -
$ cd <jalop_root>/jalop/
$ pkill jal-local-store
$ sudo rm -rf ./jal.sock    # remove old JAL socket, if any.
$ sudo rm -rf /root/testdb  # clean up existing <db_root>/ directory.
$ sudo mkdir /root/testdb   # For the first time, create <db_root>/

Make a copy of ./test-input/local_store.cfg here and update
accordingly. This is to avoid changing the sample configuration file
in the git source tree.

$ sudo ./release/bin/jal-local-store --debug ./local_store.cfg
```

6.4 Insert Records into JAL-Local-Store

```
$ cd <jalop_root>/jalop/
```

6.4.1 Insert Log Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop/test-
input/sample2.cfg -p ~/jalop/jalop/test-input/big_payload.txt -n 100 -
t l
```

6.4.2 Insert Audit Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop/test-
input/sample2.cfg -p ~/jalop/jalop/test-input/big_payload.txt -n 100 -
t a
```

6.4.3 Insert Journal Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop/test-
input/sample2.cfg -p ~/jalop/jalop/test-input/big_payload.txt -n 100 -
t j
```

6.5 Check for Inserted Records in JAL-Local-Store

```
=============================================
$ sudo ./release/bin/jaldb_tail -n 1000000 -h /root/testdb/ -t l | wc
-l
$ sudo ./release/bin/jaldb_tail -n 1000000 -h /root/testdb/ -t a | wc
-l
$ sudo ./release/bin/jaldb_tail -n 1000000 -h /root/testdb/ -t j | wc
-l

Or,
```

```
$ sudo ./release/bin/jal_purge -h /root/testdb -b 2023-11-11T11:11:11
-x -t l | wc -l
$ sudo ./release/bin/jal_purge -h /root/testdb -b 2023-11-11T11:11:11
-x -t a | wc -l
$ sudo ./release/bin/jal_purge -h /root/testdb -b 2023-11-11T11:11:11
-x -t j | wc -l
```

6.6 Start C Publisher

```
$ cd <jalop_root>/jalop/

Make a copy of ./test-input/jald.cfg here and update accordingly. This
is to avoid changing the sample configuration file in the git source
tree.

$ sudo ./release/bin/jald -s -d -c ./jald.cfg --no-daemon 2>&1 | tee
publisher.log

Enter "man jald" for help.
```

6.7 Start Java Subscriber

```
$ cd <jalop_root>/jjnl/jnl_test/

Make a copy of ./jnl_test/target/test-
classes/sampleHttpSubscriber.json here and update accordingly. This is
to avoid changing the sample configuration file in the git source
tree.

java -jar target/jnl_test-1.0.0.5.jar ./sampleSubscriber.json 2>&1 |
tee subscriber.log
```

6.8 Purge Records from Local Store

```
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2023-11-
11T11:11:11 -t l
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2023-11-
11T11:11:11 -t a
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2023-11-
11T11:11:11 -t j

OR,

$ now=$(date -u "+%Y-%m-%dT%H:%M:%S.%N")
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b "$now" -t l
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b "$now" -t a
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b "$now" -t j
```

6.9 Run the C Publisher (jald) with GDB

```
=====================================
$ export LD_LIBRARY_PATH="/home/marefin/jalop/jalop/debug/lib/"
$ gdb -ex=r --args ./debug/bin/jald -d -s -c ./jald.cfg --no-daemon
```

6.10 Run the Publisher (jald) with Valgrind

```
=====================================
$ valgrind --tool=memcheck --leak-check=full --verbose --track-
origins=yes --log-file=valgrind_out.txt ./debug/bin/jald -d -s -c
./jald.cfg --no-daemon
```

7 JALoPv1.x Data-Taps

The JALoPv1.x data-taps redirect records to the JALoP Local Store. Generally, each of the data-taps requires the JALoP `socket` and `db_root` addresses that `jal-local-store` (JLS) uses. Note that `jal-local-store` must be already running for any data-tap to send JAL records to it via the `socket`.

7.1 jalop-coreutils

Available on InteLink and JALoP development Git Server.

This has the JALoP version of GNU `tail` and GNU `tee` commands.

7.1.1 Build & Install

```
% cd <jalop_root>/jalop-coreutils/
% autoreconf -fiv
% ./configure --disable-gcc-warnings
% make -j
% make check
% [[ -e /tmp/install ]] || mkdir /tmp/install
% make install DESTDIR=/tmp/install
% cp /tmp/install/usr/local/bin/{tee,tail} /usr/local/bin
```

7.1.2 Run tee and tail

Note that the `jal-local-store` (JLS) process must be running already to receive and insert records.

```
% su -
% echo "This is a test message to tee into JLS" | tee -j --
path=<path to JALoP socket>
% echo -e "Test message1\nTest message2" > file_to_tail
$ tail -j --path=<path to JALoP socket> ./file_to_tail
```

Use `jaldb_tail` to check for newly inserted records -

```
% jaldb_tail -t l -h <db_root>
```

Use `jal_dump` to verify that the log records came from `auditd` –

```
% jal_dump -u <UUID found using jaldb_tail> -h <db_root)-t l -d
z
```

7.2 jalop-jalauditd

Available on InteLink and JALoP development Git Server. This is also available at GitHub/JALoP/JALoP-Auditd-Plugin.

7.2.1 Build & Install

```
% cd <jalop_root>/jalop-jalauditd (or JALoP-Auditd-Plugin)
% make clean
% make
% sudo make install
```

This shall install the followings –

- The binary `/sbin/jalauditd` (or `/usr/sbin/jalauditd`)
- The audisp child process configuration file `/etc/audisp/plugins.d/audisp-jalauditd.conf`
- The `jalauditd` configuration file `/etc/jalauditd/jalauditd.conf`. This file is initially empty; you will need to edit this file, see below.

7.2.2 Configure jalauditd

The `jalauditd` configuration file `/etc/jalauditd/jalauditd.conf` is initially empty. There can be 4 settings in this file as shown below.

```
socket = "/path/to/jalop/socket";
schemas = "/path/to/schemas/root";
keypath = "/path/to/key";
certpath = "/path/to/cert";
```

If the `socket` and `schemas` locations are not specified above, default locations specified by the JAL Producer Library (JPL) will be used. Below are the default socket and schemas locations –

```
socket = "/var/run/jalop/jalop.sock"
schemas = "/usr/share/jalop/schemas"
```

If `keypath` or `certpath` are not specified, no key or cert will be used for signing.

IMPORTANT: These settings must be consistent with the `jal-local-store` configuration file.

7.2.3 Run jalauditd

Note that the `jal-local-store` (JLS) process must run to receive and insert records sent by `jalauditd` via the `socket`.

Restarting `auditd` automatically runs the `jalauditd` child process.

```
% service auditd restart
```

Use `jaldb_tail` to check for newly inserted records -

```
% sudo jaldb_tail -f -t l -h <db_root>
```

Use `jal_dump` to verify that the log records came from `auditd` –

```
% sudo jal_dump -u <UUID found using jaldb_tail> -h <db_root)-t l
-d z
```

See `<jalop_root>/jalop-jalauditd/README` file for more details.

7.3 jalop-rsyslog

Available on InteLink and JALoP development Git Server.

More here …

7.4 jalop-log4cxx

Available on InteLink and JALoP development Git Server.

More here …