User Guide
JALoP over HTTP (v2.x)

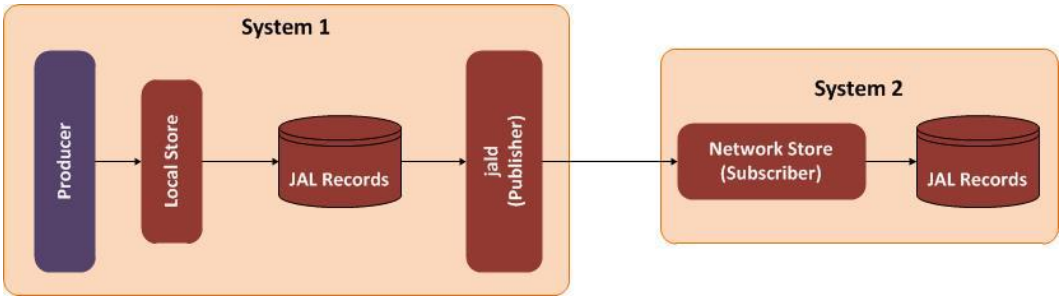
Draft

15 December 2022

Table of Contents

1 JALoPv2.x

JALoPv2.x is JALoP over HTTP. It has a C Publisher, a Java Subscriber, and several data-taps. 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 C Publisher system and “System 2” depicts a Java 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

JALoPv2.x 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/2.x.x.x (if not already on 2.x.x.x branch)

% git clone https://github.com/JALoP/jjnl.git
% cd jjnl/
% git checkout –t origin/2.x.x.x (if not already on 2.x.x.x branch)
```

2.2 JALoP CTC-Internal Git Repositories

All the JALoP source code repositories are available on the CTC-internal gitlab 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 gitlab repositories as a developer, you can git clone the repositories as below –

```
% git clone git@gitlab.cdsc.ctc.com:jalop/jalop.git
% cd jalop/
% git checkout -t origin/2.x.x.x

% git clone git@gitlab.cdsc.ctc.com:jalop/jjnl.git
% cd jjnl/
% git checkout -t origin/2.x.x.x
```

3 Build and Install the C Publisher (jald)

The C Publisher (jald) is the process that negotiates with the Java 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-scons lcov libtool-ltdl-devel libcurl-devel doxygen`
- 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`
- `Build and install axl`
  - `$ git clone` https://github.com/ASPLes/libaxl.git
  - `$ cd libaxl`
  - `$ ./autogen.sh`
  - `$ 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 --disable-py-axl`
  - `$ 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

The “jalop” (or “JALoP” if cloned from GitHub) repository has and builds the following applications –

- “jald” (the Publisher)
- “jal-local-store” (the JAL-Local-Store)
- “jaldb_tail”
- “jal_dump”
- “jal_purge”
- “jalp_test” (a test Producer)

It also builds the following shared libraries –

- “libjal-common.so”
- “libjal-db.so”
- “libjal-network.so”
- “libjal-producer.sp”
- “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 v2.x branch of jalop -
  - `% git checkout -t origin/2.x.x.x`

- Clean up and build jalop –
  - Note: PKG_CONFIG_PATH may need to be set to include /usr/local/lib/pkgconfig until we remove vortex dependency in JALoP v2.x.
  - % export PKG_CONFIG_PATH=”/usr/local/lib/pkgconfig”
  - % scons -c
  - % scons
- Install the publisher -
  - % sudo PKG_CONFIG_PATH=”/usr/local/lib/pkgconfig” ./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 negotiation
private_key = "/etc/jald/pub_key.pem";

# the path to the public cert, used for TLS negotiation
public_cert = "/etc/jald/pub_cert.pem";

# UUID used to identify this publisher
publisher_id = "cc0191c2-97e8-4cbf-af13-920d268d68ec";

# time in seconds between checks for new records when none are available
poll_time = 1L;

# time in seconds between attempts to reconnect to peers
# -1 indicates jald should not attempt reconnects
retry_interval = 5L;

# Network timeout for each session, in minutes. Upon failure to send or
receive
# data in this time, a network outage is assumed and the session closes.
# The special value of 0 implies not network timeout is enforced.
network_timeout = 25;

# path to the root of the database (optional)
db_root = "/root/testdb";

# path to a directory containing the JALoP schemas (optional)
schemas_root = "/usr/share/jalop/schemas/";

# file storing PID of jald when daemonized.
pid_file = "/var/log/jalop/jald-pid.txt";

# Log directory of jald when daemonized.
log_dir = "/var/log/jalop/log/";

# List of subscriber configurations.
peers = ( {
      # the hostname or IP address of the subscriber
      host = "127.0.0.1";
      # the port to connect to
      port = 8444L;
      # the mode of JALoP operation
      mode = "archive";
      # array of digest challenge configuration settings ordered by
descending priority
      digest_challenge = ["on", "off"];
      # array of record types to be sent to the subscriber
      record_types = ["audit", "log", "journal"];
      # directory containing the CA certificate(s) to use for TLS
negotiation
      cert_dir = "/etc/jald/remote_certs";
      } );
```

Check 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 = "./schemas/";
socket = "/home/marefin/jalop/jalop/jal.sock";
sign_sys_meta = false;
manifest_sys_meta = false;
accept_delay_thread_count = 10;
accept_delay_increment = 100;
accept_delay_max = 10000000;


# file storing PID of jal-local-store when daemonized.
pid_file = "/var/log/jalop/jls-pid.txt";

# Log directory of jal-local-store when daemonized.
log_dir = "/var/log/jalop/log/";

enable_seccomp = true;
initial_seccomp_rules =
["sched_yield","arch_prctl","bind","brk","chdir","dup2","execve","floc
k","getcwd","getdents","getdents64","getrlimit","ioctl","listen","lsta
t","poll","prctl","prlimit64","rename","rt_sigaction","rt_sigprocmask"
,"seccomp","select","set_tid_address","setsid","statfs","sysinfo"];
final_seccomp_rules =
["sched_yield","accept","access","brk","clone","close","connect","exit
","exit_group","fcntl","fdatasync","fstat","futex","getpid","getppid",
"getrandom","getsockopt","gettid","getuid","lseek","madvise","mkdir","
mmap","mprotect","munmap","open","openat","pread64","pwrite64","read",
"recvmsg","rt_sigreturn","set_robust_list","socket","stat","unlink","w
rite"];
```

4 Build and Install the Java Subscriber (jnl_test-2.1.x.x.jar)

This is the “jjnl” repository. Clone the repository as below (if not done yet) –

```
% cd <jalop_root>/

% git clone https://github.com/JALoP/jjnl.git
```

```
Or,
% git clone git@gitlab.cdsc.ctc.com:jalop/jjnl.git


% cd jjnl
% git checkout -t origin/2.x.x.x (if not already in that branch)
```

4.1 System Provisioning

Install the following packages –

```
% sudo yum install java-11-openjdk-devel ant maven
```

4.2 Build and Install the Java Subscriber

- Build the "master" branch for JALoPv2.x -
  - `% cd <jalop_root>/jjnl/jnl_parent/`
  - `% mvn clean`
  - `% mvn -Djava.version=1.8 -U clean package (for Java 8 build), or`
  - `% mvn -Djava.version=11 -U clean package (for Java 11 build)`

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

4.3 Configure the Java Subscriber

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

Copy `./jnl_test/target/test-classes/sampleHttpSubscriber.json` here and update accordingly. This is to avoid changing the sample config file in git source tree.

An example of `sampleHttpSubscriber.json` given below –

```
{
  "address": "127.0.0.1",
  "port": 8444,
  "subscriber": {
    "maxSessionLimit": 100,
    "recordType": [ "audit", "log", "journal" ],
    "configureDigest": [ "on", "off"],
    "configureTls": "off",
    "output": "./output",
    "mode": "archive",
    "createConfirmedFile" : "on",
  }
  "ssl": {
    "Key Store Passphrase": "changeit",
    "Key Store": "./certs/trust_store/server.jks",

    "Trust Store Passphrase": "changeit",
    "Trust Store": "./certs/trust_store/remotes.jks",
  }
}
```

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

5 Run Publisher and Subscriber to Transfer JAL Records

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

5.1 Disable SELinux (Test Environment Only)

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

5.2 Stop Firewall (Test Environment Only)

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


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

5.3 Start JAL-LOCAL-STORE:

```
========================================
# May need to clean up the first time. Make sure no jal-local-store is
running.
$ cd <jalop_root>/jalop/
$ pkill jal-local-store
$ sudo rm -rf ./jal.sock
```

```
$ sudo rm -rf /root/testdb
$ sudo mkdir /root/testdb

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 cource tree.

$ sudo ./release/bin/jal-local-store --debug --no-daemon -c
./local_store.cfg &

For 2.0.0.2-beta and older:
$ sudo ./release/bin/jal-local-store --debug ./local_store.cfg &
```

5.4 Insert Records into JAL-Local-Store

5.4.1 Log Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop-test-
data/input/app_meta/log4cxx-warn.cfg -p ~/jalop/jalop-test-
data/input/journal/big_payload.txt -n 100 -t l
```

5.4.2 Audit Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop-test-
data/input/app_meta/log4cxx-warn.cfg -p ~/jalop/jalop-test-
data/input/journal/big_payload.txt -n 100 -t a
```

5.4.3 Journal Records

```
=======================================
$ sudo ./release/bin/jalp_test -j ./jal.sock -a ~/jalop/jalop-test-
data/input/app_meta/log4cxx-warn.cfg -p ~/jalop/jalop-test-
data/input/journal/big_payload.txt -n 100 -t j
```

5.5 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 2033-11-11T11:11:11
-x -t l | wc -l
$ sudo ./release/bin/jal_purge -h /root/testdb -b 2033-11-11T11:11:11
-x -t a | wc -l
$ sudo ./release/bin/jal_purge -h /root/testdb -b 2033-11-11T11:11:11
-x -t j | wc -l
```

5.6 Start 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 -d -c ./jald.cfg --no-daemon -s 2>&1 | tee
publisher.log
```

5.7 Start 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-2.0.0.0.jar ./sampleHttpSubscriber.json
2>&1 | tee subscriber.log
```

5.8 Purge Records from Local Store

```
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2033-11-
11T11:11:11 -t l
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2033-11-
11T11:11:11 -t a
$ sudo ./release/bin/jal_purge -h ./testdb/ -d -f -c -b 2033-11-
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
```

5.9 Run the Publisher (jald) with GDB

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

5.10 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 -c
./jald.cfg --no-daemon -s
```

6 JALoPv2.x Data-Taps

The JALoPv2.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`.

6.1 jalop-coreutils

Available on NCDSMO Intelink SharePoint site and JALoP development GitLab Server.

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

6.1.1 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
```

6.1.2 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
```

6.2 jalop-jalauditd

Available on NCDSMO Intelink SharePoint site and JALoP development GitLab Server. This is also available at GitHub/JALoP/JALoP-Auditd-Plugin.

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.3 jalop-log4cxx

Available on InteLink and JALoP development Git Server.

More here …

6.4 jalop-rsyslog

Available on InteLink and JALoP development Git Server.

More here …