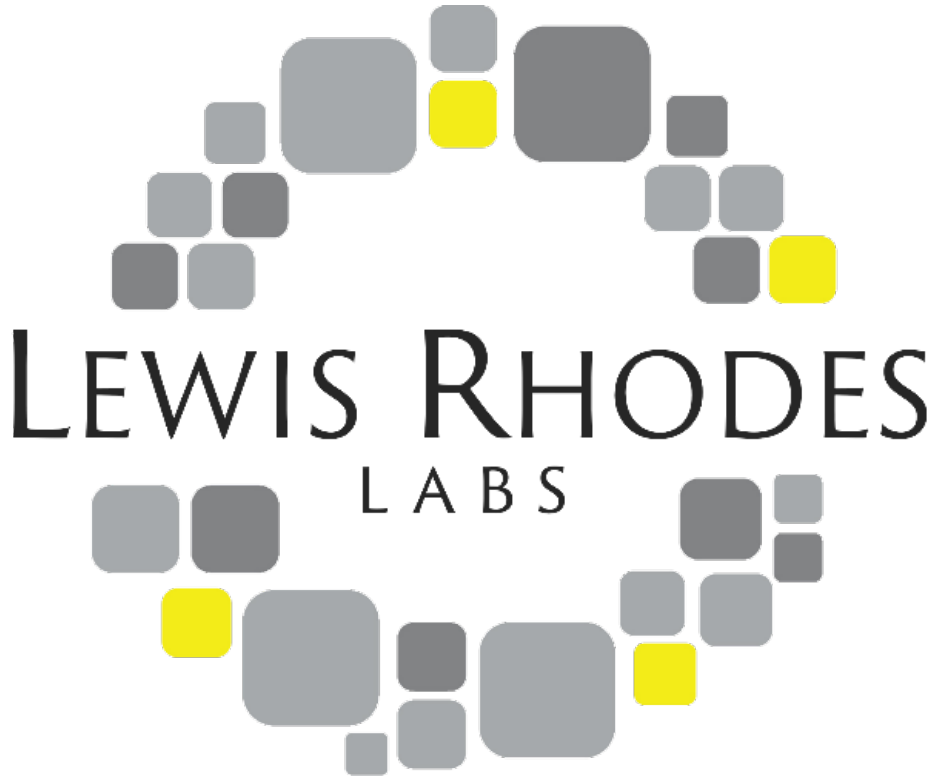

Extreme Search™ Manual

Version 2024.04

https://lewis-rhodes.com

support@lewis-rhodes.com

---

3 Configuration

3.1 General

NPU*search* itself has no fixed network port requirements. The npusearch client and backend communicate only through redis and respect the port choices that they and redis have been configured with, so long as those are consistent.

By default redis uses ports:

- `6379`

The standard required gluster ports are:

- `24007` - GlusterFS daemon
- `49152:49251` - GlusterFS bricks (this range may need to be larger if many volumes with many bricks

are hosted on a machine)

GlusterFS may require additional ports if NFS, CIFS/Samba, or other optional components are configured.

3.2 NPU*search* configuration

NPU*search* itself uses a single config file at `/opt/lrl/etc/npusearch.conf`. This config file is a simple bash script which is sourced by the npusearch service and sets environment variables.

Common keys include:

- `HOST=` the hostname of the redis server for backends on this server to connect to (default `localhost`)
- `PORT=` the port to connect to redis over (int, default `6379`, the default redis port)
- `REDIS_PASSWORD=` a password for backends on this server to use when connecting to redis, if provided (default is unset)
- `REGISTRY=` the registry name for backends to register themselves under in redis (the resulting key will be `npusearch:registry:${REGISTRY}`)
- `LOGFILE=` path, file to use for logging in addition to journalctl (default `/dev/null`)
- `RETRY=` int, number of times to retry connecting to redis before the service gives up
- `RETRY_WAIT=` int, seconds to wait before rechecking redis on startup if redis can't be reached

Less common keys include:

- `RLM_LICENSE=` license file to check (other environment variables that control the RLM licensing software may also be used)
- `MATCH_GAPS=` the default number of match_gaps for backends to run if not overriden in a request (int, default=16, current hardware maximum is 16 on all platforms, will be clamped to a valid value)
- `CHAINS=` the default number of chains for backends to run if not overriden in a request (int, default=16, maximum on SmartSSD systems is 12 and on Kuona systems is 16, will be constrained to a usable value)

Debug/very special configuration only keys include (**YOU PROBABLY DON'T NEED TO SET THESE**):

- `PROG=` path, the program to run for the backends, defaults to the platform default.
- `SBUFF_SIZE=` int, size in bytes of metadata buffers on the hardware for the backends to try to use.
- `DBUFF_SIZE=` int, size in bytes of data buffers on the hardware for the backends to try to use.

Kuona only:

- `POLL_USEC=` int, poll interval when waiting for the hardware, in microseconds.
- `POLL_MSEC=` int, poll interval when waiting for the hardware, in milliseconds.

SmartSSD only:

- `XLA_PLATFORM=` string, name of xrt the shell to use.
- `XCL_BIN=` path, location of the xrt shell to use.
- `TASK_COUNT=` int, depth of pipelining for the backends to use. Will be clamped to an allowed value.

3.3 Redis configuration

Redis is generally configured with a configuration file at `/etc/redis/redis.conf` although if redis is run in some other way that may vary. See the redis documentation for more details
https://redis.io/docs/management/

The important things for redis configuration are that:

- every backend **MUST** be able to talk to redis
- the frontend/client **MUST** be able to talk to redis
- currently the backends communicate with redis via `libhiredis`, which does not support TLS connections, so redis **MUST NOT** configured with TLS (This will likely change in the future. Redis may also not be used in the future.)

- clients and backends currently communicate via redis `PUBSUB`. If requests will be made with very large encoded size (i.e., very long literal lists of files or lists of globs, rather than globs which match many files) it is likely that the `client-output-buffer-limit` for `pubsub` will need to be increased. Additionally, if there are many clients making requests concurrently this value should be increased. The redis default is 64MB of buffer at any one time, or 8MB of buffer for at least 60s, but the soft and hard limits **SHOULD** be raised to allow at least 1MB of buffer per anticipated concurrent backend. (In a future release communication is expected to move away from `PUBSUB` primarily, at which point this will be relaxed). If scans are returning sporadic "no one is listening" errors under highly concurrent use, `client-output-buffer-limit` being too low should be suspected.

Note: npusearch namespaces the keys and pubsub channels it uses by prefixing them with `npusearch:`. If keys and channels with these names are not being used elsewhere, the redis **MAY** be shared with other uses. If backends for different registries are on hosts with different hostnames, multiple npusearch clusters **MAY** use the same redis server.

3.4 GlusterFS configuration

If GlusterFS is being used for file placement there are some specific requirements needed to make use of the gluster/npusearch integration. By default gluster is configured using the `gluster` commandline utility. Using some other glusterfs management system may cause this to vary. GlusterFS documentation can be found at https://docs.gluster.org/

Specific requirements for GlusterFS to work with the npusearch integration currently. (Some of these will likely be relaxed in the future)

- gluster volumes **MUST** be configured as "distributed" or "distributed replicated". "dispersed" volumes **MUST NOT** be used.
- gluster volumes **MUST** be backed by bricks on npusearch SmartSSDs
- npusearch SSDs **MUST** be formatted with a file system which supports `O_DIRECT` (e.g., XFS, ext4). At present the partitions **MAY** be on LVM logical volumes but each logical volume **MUST** only be backed with PVs from a single npusearch drive.
- if gluster integration is desired the backends **MUST** be run with permissions to read the files in `/var/lib/glusterd/vols/**`