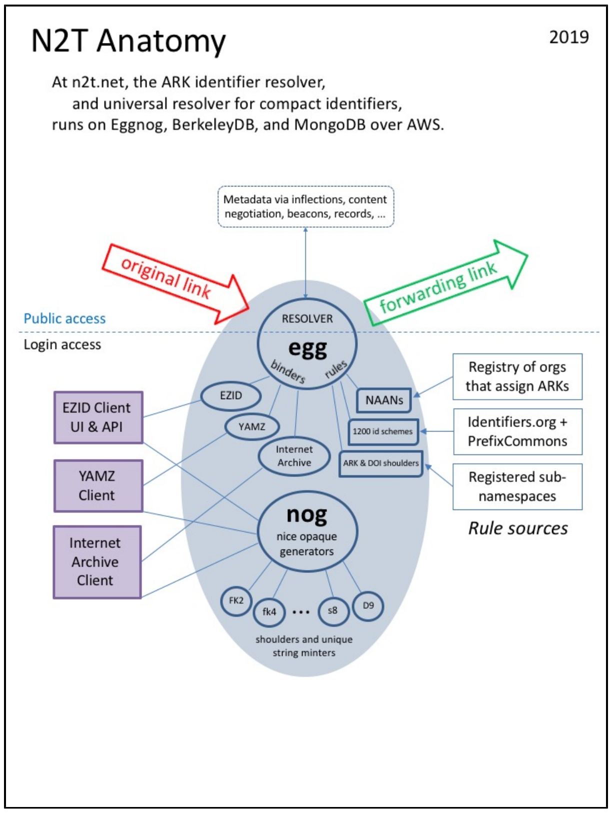
N2T Operations

Contents


N2T Anatomy
2019
At n2t.net, the ARK identifier resolver, and universal resolver for compact identifiers, runs on Eggnog, BerkeleyDB, and MongoDB over AWS.
Metadata via inflections, content negotiation, beacons, records, ...
original link
forwarding link
Public access
Login access
RESOLVER
egg
binders
rules
EZID
YAMZ
Internet Archive
NAANs
1200 id schemes
ARK & DOI shoulders
EZID Client UI & API
YAMZ Client
Internet Archive Client
Registry of orgs that assign ARKs
Identifiers.org + PrefixCommons
Registered sub-namespaces
Rule sources
nog
nice opaque generators
FK2
fk4
s8
D9
shoulders and unique string minters

Overview

N2T provides minting, binding, and resolution services with ARK and DOI identifiers for several clients, the most important of which is the EZID service. While N2T supplies its primary backend storage, EZID is managed as a separate, independent service (see EZID Operations --secured). Other clients, used mainly for resolution, include the Internet Archive's Open Content Alliance (OCA) and the YAMZ metadictionary. Per an MOU with the EMBL-EBI, on a daily basis N2T harvests "prefix" records from Identifiers.org to support "compact identifiers".

All functions are served from an Apache web server homed at n2t.net. A replica web server (a read-only mirror) running at EDINA in Edinburgh, Scotland is updated from the primary N2T server on a regular basis via "cron".

The N2T web server runs under Amazon Web Services (AWS). A typical instance is an Amazon Linux AMI using the open-source Eggnog package (formerly Noid) to mint, bind, and resolve identifiers. Egg is for binding (storing) and resolving, and nog (nice opaque generator) is used for minting. Eggnog is written in Perl and uses the open-source BerkeleyDB software as a fast, scalable database. Resolution is provided by Apache server rewrite rules hooked up to egg binders. Also on N2T are administrative scripts written in Bash for creating and managing identifier prefixes (shoulders), backing up the database, updating replica servers, and documents to support these functions.

All the above functionality is installed identically on the EDINA replica under a RedHat Linux VM at n2tlx.edina.ac.uk, but all functions are disabled by policy flags, with the exception of resolution. Furthermore, no traffic going to n2t.net will be served by n2tlx unless and until EDINA staff are authorized to advertize n2t.net as a synonym for n2tlx. Therefore, the EDINA replica is currently inactive except to those who know how to demo its resolution function.

Instances

There are several instances of N2T running at CDL in Oakland and Berkeley, California, as well as a replica N2T service in Edinburgh. While most instances are running under Linux, there is a legacy instance and some support scripts still running under Solaris. The Amazon production and stage URLs each point to an AWS EC2 instance.

The Linux environments in which these instances run (the filesystems, role accounts, permissions, package dependencies, etc.) are mostly identical except for slight configuration differences as indicated in the table below. Effort is made to ensure that all instances are running the same version of the N2T code, but differences do occasionally creep in (as when, for example, a change has not yet been pushed to production).

	primary	replica	staging	development
N2T API and UI	n2t.net internal: ids-n2t2-prd.n2t.net	n2t.edina.ac.uk	n2t-stg.n2t.net internal: ids-n2t2-stg.n2t.net	n2t-dev.n2t.net internal: ids-n2t2-dev.n2t.net
updated by API	yes	no	yes	yes
updated by cron	no	yes	no	no
binder backups	yes	no	no	no
runs EZID account setup	no longer	no	no	no

The name "staging" above is a little misleading, as it is more like a second development instance.

To bring up N2T "from scratch" on a brand new host, create an empty director ~/n2t_create, copy in the ec2_bootmake script, and run it.

Common filesystem layout

All instances of N2T have the same filesystem layout, although there are certain directories and binders (databases) that are used regularly only on the primary instance. For example, the production directories that support shoulder creation, replication, backups, and logging are used only on the production instance; except for keeping the binders up-to-date on the replica instance, no other attempt is made to synchronize such directories. The result is that, on non-production instances, binders and directories designed to hold data, are often vestigial or contain test data of no lasting value.

In the filesystem names below, a terminal '**/**' indicates a directory and the color **red** indicates a symbolic link.

- **~n2t/** (role account home directory)
  - **/init.d/apache** (service control script)
  - **sv/cur/** aka **$sv** (service version: current; contains all system dependencies -- type "svu" to read more)
    - **apache2/** aka **$sa** (root of main Apache web server, all built by **build_server_tree**)
      - **binders/, minters/, shoulders/** (binders, minters, and shoulders, symlinked to from home directory)
      - **prefixes.yaml** (critical file with overview of all prefixes (schemes, NAANs, shoulders) known to N2T resolution)
      - **htdocs/a/** (root of all API calls; examples below)
        - **ezid/b** (link to script servicing API calls to the "ezid" main binder)
        - **ezid_test/b** (link to script servicing API calls to the "ezid" test binder)
        - **ezid/m** (link to script servicing API calls to "ezid" minters)
        - **ezid/rmap_ezid** (long-running script that resolves from the "ezid" main binder)
      - **htdocs/e/** (root of all extras, eg, documentation)
      - **logs/access_log.*YYYY.MM.DD*** (logging web accesses, including resolution)
      - **logs/transaction_log.rlog** (logging unified view of N2T system activity, minus resolution)
      - **logs/rewrite_log.*YYYY.MM.DD*** (used mostly for debugging, eg, resolution)
      - **pfx_work/import** (staging for ~/pfx_harvest components)
    - **bin/** (all commands that N2T depends on)
      - **egg** (binder commands -- type "egg" to read more)
      - **nog** (minter commands -- type "nog" to read more)
      - **db_dump, db_hotbackup, db_verify,** etc (Berkeley DB utilities)
      - **perl, python, hg, anvl, pt,** and many other commands
    - **build/** (sources for all dependencies, all built by **n2t_create/make_instance**)
    - **build/eggnog/** (an hg repo: main system sources for **egg, nog, build_server_tree** and more)
  - **sv/new/, sv/old/** (other service version symlinks)
  - **sv/cv0/, sv/cv1/, ... sv/cv9/** (actual service "code" versions, pointed to by cur, new, and old symlinks)
  - **binders/** (all binders in the current service version)
    - **ezid/** (main binder for the EZID populator)
      - **egg.bdb** (binder BerkeleyDB database for the EZID populator)
      - **egg.rlog** (replication log file, harvested periodically into **egg.orlogs/to_ship/**)
      - **egg.orlogs/** (old replication log files)
        - **to_ship/** (replication log files awaiting shipment to replica)
        - **shipped/** (replication log files that have been shipped to replica)
        - **archived/** (older, compressed, previously processed replication log files)
        - **to_play/** (replication log files received by replica instance, awaiting playback to update the replica)
        - **played/** (replication log files on replica instance that have been played back)
      - a number of ancillary files
    - **ezid_test/** (test binder for the EZID populator)
    - **n2t/** (main binder for the N2T populator -- under development to hold NAANs and shoulders)
    - **oca/** (main binder for the OCA populator)
    - **yamz/** (main binder for the YAMZ populator)
  - **minters/** (all minters in the current service version)
    - **ezid/ark/** (minters for all of EZID's ARKs and DOIs)
      - **13030/** (shoulder minters for CDL ARKs)
      - **b5072/** (shoulder minters for 10.5072 DOIs)
        - **fk2/** (minter directory for fake DOIs)
          - **nog.bdb** (minter BerkeleyDB database for fake DOIs)
      - **c3010/** (shoulder minters for 10.13010 DOIs
      - ... and many other NAANs with minters
    - **oca/ark/** (shoulder minters for Internet Archive ARKs)
      - **13960/t/** (minter for OCA ARKs)
    - **yamz/ark/** (shoulder minters for YAMZ ARKs)
      - **99152/h/** (minter for YAMZ ARKs)
  - **naans/** (all ARK NAAN management in the current service version)
    - **master_naans** (text file containing NAAN registry)
    - **forge/** (NAAN set up scripts)
      - **naan_rec** (script to generate a NAAN and registry entry)
    - **form2naa** (script to convert emailed form response to NAAN entry)
  - **shoulders/** (all shoulders in the current service version)
    - **ezaccts/** (EZID account set up requests)
    - **install-shoulders** (script to copy shoulder database to where EZID picks it up)
    - **master_shoulders.txt** (text file containing shoulder "database")
    - **reload-all** (script to cause EZID instances to pick up new shoulder database)
    - **validate-shoulders** (script to run various checks on shoulder database)
  - **backups/** (regular binder backups)
  - **logs/** (collection of mostly symlinks, currently deprecated – see **~n2t/sv/cur/apache2/logs/** above)
  - **local/bin/** (production versions of scripts maintained in **n2t_create**)

        - **`replicate`** (run from crontab to update the replica instance)
        - **`svu_run`** (implements the "svu" bash function, setting PATH, prompt, etc)
        - **`wegn`** (command line web client for exercising this N2T instance -- type "wegn" to read more)
    - **`n2t_create/`** aka **`$sn`** (an hg repo: source for scripts that maintain the n2t role account and system, and build both from scratch)
        - **`make_instance`** (script to rebuild some or all system dependencies, with "**`svu new`**" in effect)
        - **`ec2_bootmake`** (script to build new system from scratch, finishing with call to "**`make_instance build all`**")
        - **`skel/`** (source for establishing role account from scratch: basic directories, startup files, generic warts files)
    - **`pfx_harvest/`** (incoming prefix, NAAN, shoulder harvests)
    - **`README`** (summary of role account home directory)
    - **`ssl/`** (certificate files referenced by **`warts/env.sh`** environment variables)
    - **`warts/env.sh`** (sets instance-specific environment variables)

The **`~n2t/.bashrc`** file defines two critical things:

- **`PATH=$HOME/local/bin:$PATH`**
- **`function svu { eval `svu_run "$PS1"\|\|\|b "$@"`; }`**

When "**`svu cur`**" is in effect, `$PATH` is modified by prepending the current service "bin" directory to it, and `$PERL_INSTALL_BASE` is similarly modified. Other useful things are defined as well, including

```
# Some aliases that make n2t/eggnog development and testing easier.
#
alias blib="perl -Mblib"
# $PERL_INSTALL_BASE interpolated at run time, eg, when "svu cur" in effect
alias mkperl='perl Makefile.PL INSTALL_BASE=$PERL_INSTALL_BASE'
```

Administration: starting & stopping

Administering N2T is done while logged in to the "n2t" role account. The following script provides the basic controls:

```
~/n2t/init.d/apache start|stop|restart|status
```

Another script, `/etc/init.d/apache.n2t`, calls this script on machine boot and shutdown.

For maintenance downtimes that you expect to last (or that end up lasting) over a minute, it is best to suspend Nagios (see Administration: monitoring) using, for example, the "scheduled downtime" menu selection, so that other staff won't be inconvenienced by unnecessary alerts.

Administration: server status

The status of the production N2T server can be viewed by logging in via ssh to n2t@ids-n2t2-prd.n2t.net and running

```
$ ~/init.d/apache status
success: N2T is up
```

The underlying API method is described in the API documentation, which can be viewed at The N2T API.

Administration: resolver status

Information about a long-running Perl process that handles all ARK resolution can be viewed from any browser by trying to resolve `ark:/99999/__rrminfo__`. This is a special ARK (an easter egg), that asks for information from the binder via "rewritemap resolver mode" (the long-running process), and it returns a fake URL destination. For example, to query the production instance,

```
$ wget -O - n2t.net/ark:/99999/__rrminfo__ 2>&1 | grep Location
Location: http://www.example.com/embrpt/1.00/1.83/4.007025/4.7/11598/2015.02.21_23:11:51?\
dvcsid=9e1430c24de8&rmap=/apps/n2t/sv/cv2/apache2/htdocs/a/ezid/rmap_ezid [following]
```

Entered into a regular browser, the special ARK will appear to fail, but the actual information you want will be conveyed in the text of the URL displayed in the location bar. In the URL path will be found version numbers for (using "embrpt" as mnemonic) **e**gg, BerkeleyDB **m**odule, **b**uilt with which BDB C library, and **r**unning dynamically linked against which BDB C library, followed by **p**rocess id and build **t**ime. In the query string will be found the changeset id (dvcsid), and script name (rmap).

Administration: pausing the server

While there is currently no straightforward way to pause N2T, there are some actions (from the n2t role account only) that effectively pause all write operations on a binder, such as when a backup is done ("`egg -d ezid dbsave ...`"). You can artificially pause a binder from the role account from a terminal window by running, for example, "`egg -d ezid --lock dbinfo all | more`", leaving the "`more`" pager on the first page, and unpause it by quitting the pager.

Administration: patching the operating system

Installing operating system updates (often needed for security reasons) consists of a patching step, an optional machine reboot step, and a testing step.

1. *Patching step*. Under AWS patching is done with "`admegn sysupdate`", which ultimately calls "`sudo yum update`".
2. *Rebooting step.* If a reboot is required, the safest way to prepare for it in the current production configuration, is to start no sooner than 1 minute after the beginning of the advertised maintenance window, giving customer processes that observe the same window a chance to finish pausing. In addition, because the production instance is a single EC2 instance, it is best to avoid system downtime when an EC2 healthcheck comes in. So on the production host, use "`admegn pingwhen`" to observe the periodicity of the monitoring health checks (look for "WebInject") and time your reboot to happen just after the next check. If a check comes in when the system is down, the greater the chance of the instance being taken out of load balancer rotation" and the greater the apparent downtime to users. The following reboot procedure, which relies on several open terminal windows, is advised.

   - window A: ssh session into production
   - window B: a non-production machine (eg, laptop) from which you will initiate the reboot

   a. To start (if patching, do that first, then), from window A understand when the healthchecks occur. Note that by current local AWS policy, the "WebInject" healthchecks may be suspended during EZID's standard maintenance period; if so this step may not be necessary (also reboot time has shortened recently from 90 to 20 seconds, which further reduces concerns).
   ```
   admegn pingwhen                                     # look for the timing of "WebInject" checks
   admegn pingwhen                                     # verify latest check
   ```
   b. From window B, initiate and monitor the reboot, to let you know when the system is back up and how long it was down.
   ```
   admegn bootwatch ids-n2t2-prd.n2t.net n2t.net       # reboots remote system
   ```
   Keeping an eye on window B during this time, verify that the ssh session in window A has disconnected.

   Testing step. An important test of whether EZID is working is to create a demo ARK at ezid.cdlib.org from a browser window and watch the `ezclp ... monitor` output until the binder queue drains (you can use the "eztest" CLI script for this instead). If that test fails, try to visit n2t.net from a browser window, and log back in to production in window A to run post-reboot tests from the shell.

   a.
   ```
   n2t test                                            # test script (some failed tests may be normal)
   ```

Administration: monitoring

There are several sources of system monitoring. All sources notify of outages but take no action themselves.

1. From CDL, monitoring of CDL hosts is managed by Nagios, with email notification going to the developers' email addresses.
2. From EDINA, monitoring of the replica is done via a proxy (a load balancer). With only one VM actually running the replica at "n2tlx.edina.ac.uk", the proxy selects the VM if it is up, and redirects to a standard EDINA error page otherwise. Email notification goes to n2t@cdlib.org which forwards to the n2t role account, where the ~/.forward file sends it on to relevant staff.
3. From monitor.us (a free monitoring service), outage notifications for the primary and replica instances are sent to jak@ucop.edu.

For CDL monitors, action is required because CDL and UCOP staff pay close attention to service faults and will take steps to restart the service themselves unless you formally "acknowledge" the faults.

An easy way to acknowledge a fault is to reply to the emailed Nagios notification, but this must be done from Outlook mail (eg, not gmail), making the first line of the reply of this form:

```
ACK I am working on this. -John ENDACK
```

Alternatively, you may login to Nagios at https://cdl-aws-nagios.cdlib.org/ with your dev/stg (UCOP CDL LDAP) credentials, select Hosts from the left pane, select the specific host that faulted (n2t*), and select "Acknowledge..." from the Service Commands pane.

If a service is purposely taken offline, communicate that fact by formally per instructions documented in the CDL Nagios User's Guide.

Administration: maintaining the ARK NAAN registry

To get started, go into the ~/naans directory, type "./form2naa", and follow the instructions there. This directory has its own Bitbucket repository (this section is a work in progress)

Administration: setting up and tearing down shoulders and EZID accounts

To get started, go into the ~/shoulders directory, type "make_ezacct", and follow the instructions there. (this section is a work in progress)

To tear things down, a number of procedures are documented here: Transferring a client from EZID to another provider* .

Administration: contacts and after-hours IT help

For after-hours help with the CDL instances, contact the UCOP ITS Helpdesk at helpdesk@ucop.edu or 510-987-0349. If urgent help is needed to correct a serious problem, follow this procedure.

- Call 510-987-0349 to describe the problem, ask the person to contact the "UNIX On-Call Technician", and have them call you at a number where you can be reached.
- After hanging up, call Kurt Ewoldsen (CDL's IAS Service Manager) at 510-812-6635 to tell him about the problem.
- After hanging up, send email to helpdesk@ucop.edu and Kurt.Ewoldsen@ucop.edu to create a written record of the contact.

For questions about the EDINA instance, contact edina.infrastructure@ed.ac.uk (as of writing, this may reach david.elsmore@ed.ac.uk or matt.hodson@ed.ac.uk).

Source code

The source code is managed using Mercurial. There are eight central source code repositories located at bitbucket.org: one for the eggnog package, one for system and role account initialization (n2t_create), one for performing service roll out/back (svu), one for NAAN and shoulder management (NAANager), and four for separate Perl modules (newer than their CPAN counterparts). To access a repository, you'll need a personal Mercurial /RhodeCode account on hg.cdlib.org, and your account must have been granted access to the repository by the administrator (currently Martin.Haye@ucop.edu).

- eggnog repository
  - Purpose: main software (Perl and Bash) behind the N2T service
  - Central repository URL: ssh://hg@bitbucket.org/cdl/n2t-eggnog
  - Local filesystem: **`~/sv/cur/build/eggnog`**
  - Description: egg, nog, and anvl2egg commands, plus various Perl modules, config files, Bash scripts, and numerous test suites

- n2t_create repository
  - Purpose: automated N2T system build and initialization
  - Central repository URL: ssh://hg@bitbucket.org/cdl/n2t-admin
  - Local filesystem: **`~/n2t_create`**
  - Description: creates system software stack and role account from scratch
- naans repository
  - Purpose: record of NAAN registry changes
  - Central repository URL: ssh://hg@bitbucket.org/jak/naans
  - Local filesystem: **`~/naans`**
  - Description: global ARK registry, drives N2T resolution for external resolvers

- svu repository
  - Purpose: automated Service Version Utility for service rollout and rollback
  - Central repository URL: ssh://hg@bitbucket.org/cdl/svu
  - Local filesystem: copied to **`~/n2t_create`**
  - Description: service version utility for rollout and rollback

- shoulders repository
  - Purpose:
  - Central repository URL: ssh://hg@bitbucket.org/cdl/shoulders
  - Local filesystem: **`~/sv/cur/shoulders`**
  - Description: NAAN and shoulder management

- File-ANVL, File-Namaste, File-Pairtree, File-Value repositories
  - Purpose: general-purpose supporting Perl modules (newer than their http://search.cpan.org/~jak/ counterparts)
  - Central repository URLs: ssh://hg@bitbucket.org/cdl/File-...
  - Local filesystem: **`~/sv/cur/build/File-...`**

At system build time, N2T populator (ie, user) passwords are looked for first in the file **`~/warts/.pswdfile.n2t`** and then in **`~/sv/cur/build/eggnog/t/n2t/pswdfile`**. Do *not* put real passwords in the latter file, which is (a) checked into the source repository and (b) meant to be used only for test purposes.

The linkage between a repository in one of the N2T instances and its counterpart on `hg.cdlib.org` is described in **`<repo>/.hg/hgrc`**. As a consequence, it's possible to push and pull without specifying the central repository, e.g.:

```
cd ~/sv/cur/build/eggnog
hg push|pull|incoming|outgoing|etc.
```

Command-line tools

The `egg` command manages binders and the `nog` command manages minters. To see the state of any binder or minter, use the `mstat` sub-command, as in

```
$ nog -d ~/minters/ezid/ark/99999/fk4 mstat
minter: /apps/n2t/minters/ezid/ark/99999/fk4
status: enabled
spings minted: 852651
spings left: unlimited
spings left before blade expansion ("add3"): 69883859
spings held: 0
spings queued: 0
template: 99999/fk4{eedeedk}
sample: 99999/fk40002f9j
original template: 99999/fk4{eedk}
$ egg -d ~/binders/ezid mstat
binder: /apps/n2t/binders/ezid
status: enabled
bindings: 58623062
```

Other command-line tools (in ~/local/bin) provide a variety of services. To print help information about how to use the tools below, at a shell prompt type the name of the tool with no arguments.

- **`admegn`** `- admin tool for eggnog, eg,` **`admegn backup`**
- **`mg`** `- mongodb server and result set management`
- **`pfx`** `- prefix harvesting, testing, installation`
- **`wegn`** `- web eggnog client`
- **`ezcl`** `- web EZID client`
- **`wegnpw`** `- fetch an N2T user's password`
- **`n2t`** `- admin tool for the N2T service, eg,` **`n2t rollout`**
- **`bdbkeys`** `- fetch keys and values from one or more Berkeley DB databases`

Procedure: code update

Imagine some code changes have been committed and pushed to hg.cdlib.org from dev. To deploy those changes in stage:

1. Login to n2t@ids-n2t2-stg.cdlib.org.
2. **`cd $sv/build/eggnog`**
3. **`n2t import`**
4. **`n2t rollout             # about .5 seconds downtime`**
5. **`n2t test`**

If something goes terribly awry, to rollback to the previous revision:

1. **`n2t rollback`**
2. **`n2t test`**

If everything looks good, repeat the previous steps on the other N2T instances, ending with the production instance.

If you need to install a new version of a major dependency, such as the Perl interpreter, you will want to use "**`svu new`**" to install an entire new service version.

1. **`cd ~/n2t_create`**
2. **`svu new`**
3. **`vi make_instance              # to change the Perl version to be installed`**
4. **`./make_instance build all     # 15-25 minutes to build full stack of software dependencies and test thoroughly`**
5. **`... svu rollout ...           # not all details have been tested`**

Cron jobs

A cron job runs every 3 minutes on the primary production instance to update the replica by calling **`~n2t/local/bin/replicate`**. Note that replication as currently implemented can cause a crippling load to the EDINA replica when there are very large numbers of updates, such as when batch changes are occurring; in such cases it may be best to disable replication by commenting out the above line using **`crontab -e`**, to copy the database manually to EDINA when all updates are done, and then to restart replication.

Another regular maintenance task is the manual weekly backup of the EZID binder. This is complemented by daily backups of a separate EZID client-side database.

Passwords, populators, and ports

Information on passwords, populators, and ports is maintained in a separate document.

Personally identifiable information and privacy

N2T stores opaque "agent PIDs" recorded by EZID. Agent PIDs are viewable only by the administrator.

In its role as maintainer of the NAAN registry for ARK identifiers, N2T stores contact information (names, email addresses, physical addresses, affiliations, phone numbers) associated with NAANs in ~/shoulders/naans/master_naans. Contact information is viewable only by the administrator and is encrypted (lightly) before the registry is replicated (only to sites that do not themselves make such information available).

In its role as maintainer of the EZID shoulder database, N2T stores some EZID login account names in ~/shoulders/master_shoulders.txt. Contact information is viewable only by the administrator.

The identifier operations that a populator (a service, such as EZID, OCA, or YAMZ, not an individual) performs are recorded in N2T's transaction log. Transaction log records are retained indefinitely and are viewable only by N2T software development staff.

The Apache access log includes client/browser IP addresses. Access log records are retained for six months.