The Near-line Data Store

Bridging the gap between disk and tape

Neil Massey – neil.massey@stfc.ac.uk

Thanks: Jack Leland, Bryan Lawrence, William Cross, Nicola Farmer

Overview

1. Current tape systems available to JASMIN users, and their problems.
2. The Near-line Data Store architecture.
3. The Near-line Data Store as a user.
4. Conclusions and next steps.

Science and Technology Facilities Council
Natural Environment Research Council

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

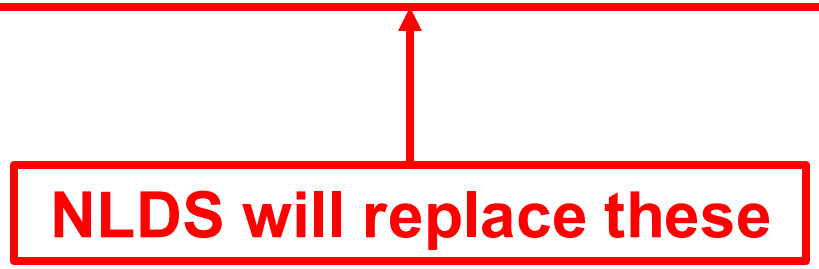
Tape, in 2025?

- Tape might seem old-fashioned, but it retains relevance
- Data volumes are increasing
- Increasing resolution of climate models, satellite instruments, etc.
- Increasing number of experiments, measurements, etc.
- Pressure to reduce energy usage from a cost and Net-Zero perspective
- Tape offers solution: cheaper per TB than disk to purchase, and run
- Uses less energy, especially when dormant
- Less e-waste on decommissioning
- But, to be used effectively, it requires a user-friendly interface

Current JASMIN tape systems

- Near-line Archive (NLA) – for CEDA archive data that is too large to store on disk. Read only.

- Elastic Tape (ET) – for group-workspace managers to ingest data from group-workspaces to tape.
- Joint Data Migration Application (JDMA) – A “user-friendly” wrapper on top of ET that catalogues data, allows migrations and reports the status of jobs.



Problems with ET

- Only available to GWS managers and their deputies.
- Direct interaction with the tape, leading to contention and ingest errors.
- Only one version of a file can be stored in ET. Precludes any iterative backup.
- Can be difficult to monitor progress of a job – only viewable via a website.
- Data is catalogued but not searchable. Only viewable via the website.
- Data retrieval is synchronous. Error requires whole retrieval to be restarted.
- No authentication or authorisation!
- No way to check quota.

Problems with JDMA

- JDMA is a wrapper for ET
- Inherits all of ET's problems and adds some of its own.
- Directory locking has been particularly problematic for some users.
- Cataloguing system is an improvement over ET – but there are now two catalogues of user's data (ET and JDMA)!
- Still no authentication
- Still only for GWS managers
- Historically poor performance

The Near-line Data Store

NLDS is a hierarchical file management system

- Users can ingest files from hot (disk) to warm (S3 object) storage
- Data is catalogued on ingest
- Data is automatically backed up to cold (tape) storage
- Data might be removed from warm storage via policies
- Retrieval from cold or warm storage via same command
  - Only difference from user point of view is the time lag
- Ingest and retrieval are asynchronous
- No proprietary formats for storing the data

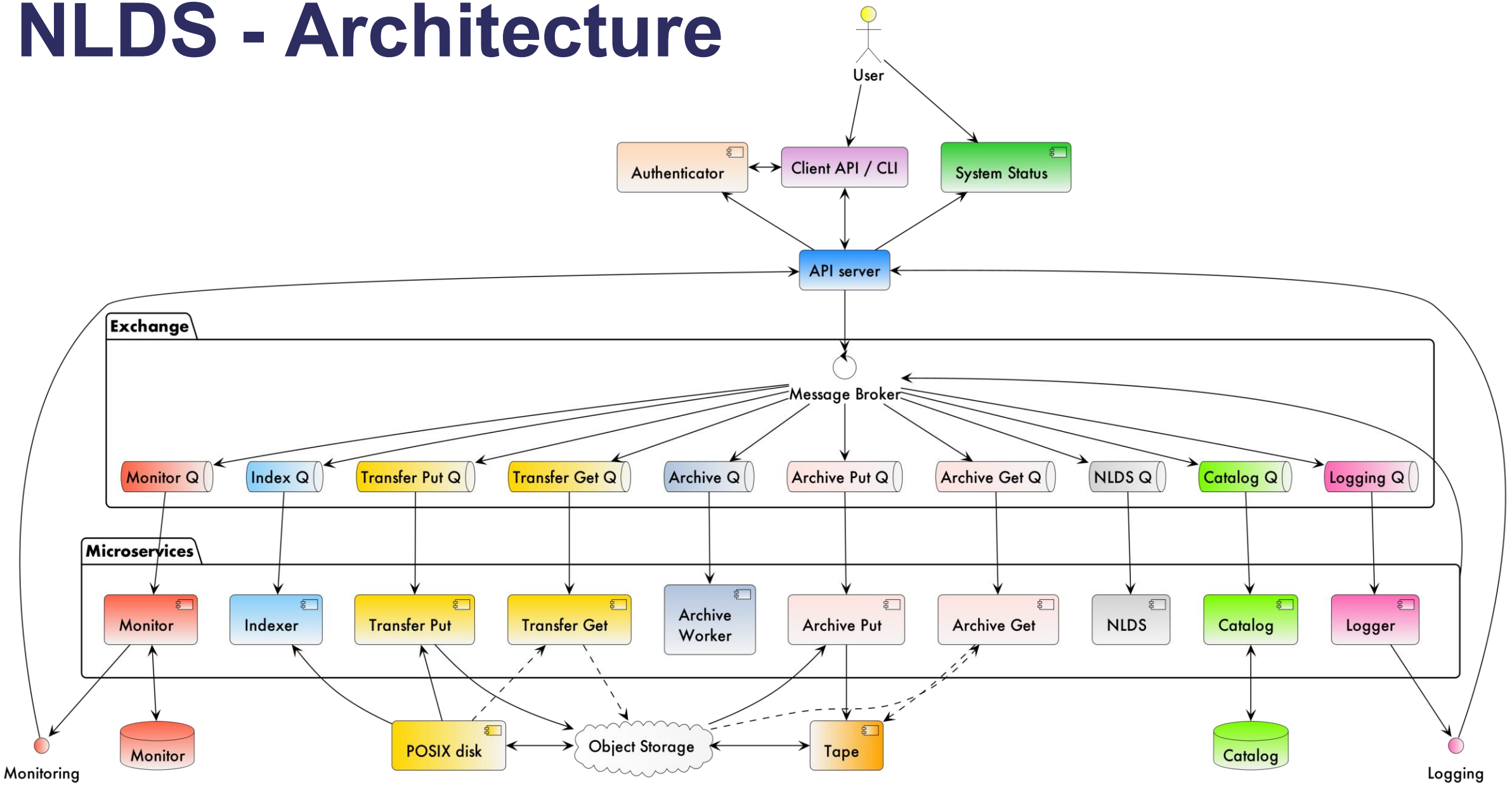
NLDS - Design

- RESTful API
  - Client library connects to API, client program uses client library
  - API is published in OpenAPI via FastAPI
- Portable – can be installed on any multi-user system
  - Use any S3 object storage on premise
  - Or remote – e.g. AWS remote storage
- Available to all users, not just GWS managers / deputies
- Implements CRUD (create, read, update, destroy)
- RabbitMQ message broker
- Open Source (of course)


NLDS - Technology

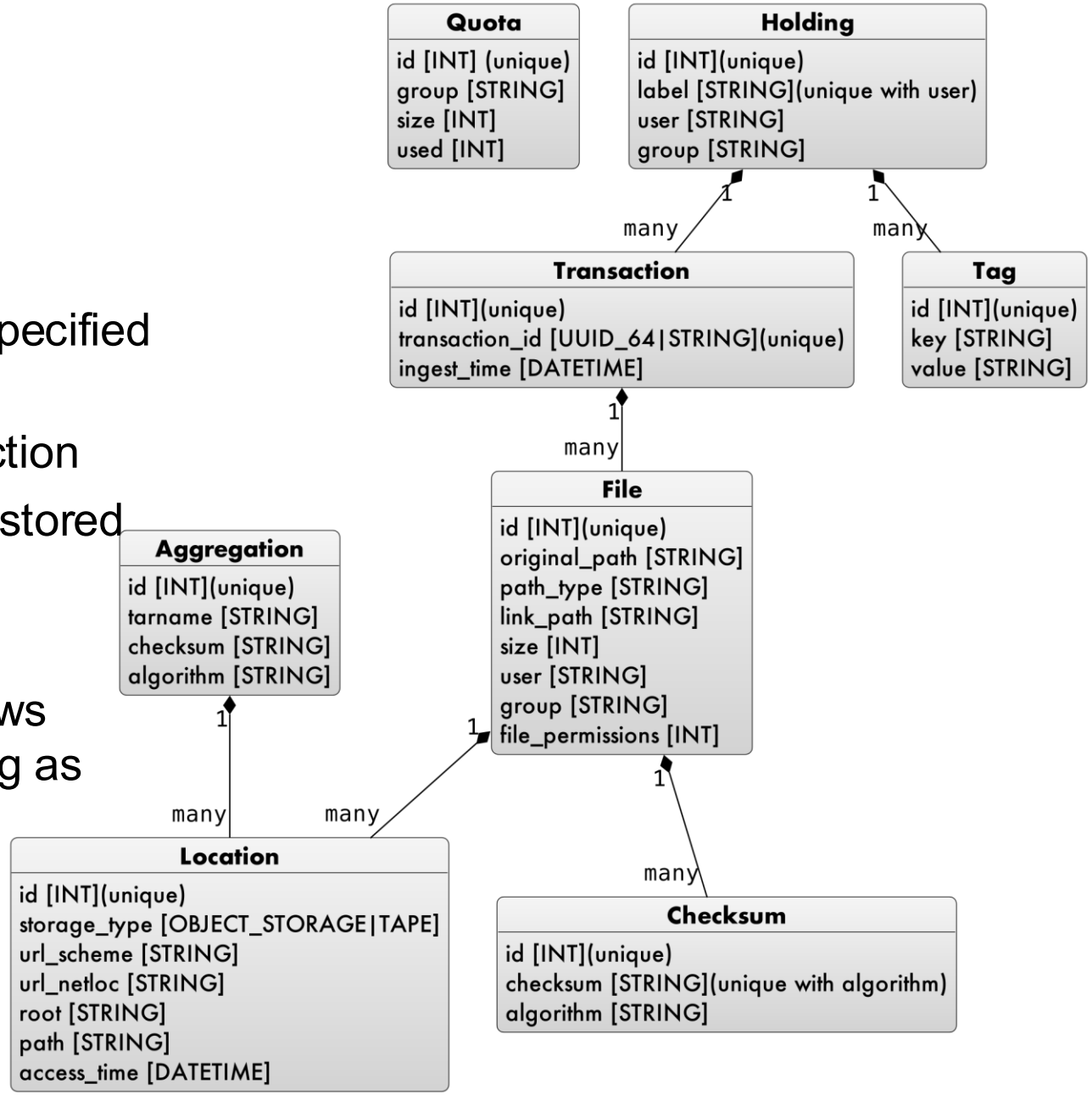
- NLDS uses **OAuth2** to determine what a user can do
  - Client library handles user's token storage and requests to NLDS
  - Server has custom "plug in" class for JASMIN authentication
  - Portable: Can write a class for another authentication system (Apple, Google, etc)
- **Rabbit MQ** used as the message brokering system:
  - Asynchronous, parallelism of tasks, resending of lost messages
  - Topic-queue routing allows microservice specialisation and re-use
- Server-API and microservices deployed on containers via **Kubernetes**
  - Portable, zero-downtime config changes and updates
  - Dynamic redeployment under load
  - Deployed via Helm charts and GitLab continuous integration

NLDS - Architecture



NLDS - Catalog

- Files can be added to a holding at any time
- Holdings are identified by a label, which are constrained to be unique for a user
- A new holding is created if a holding with the specified label is not found
- Default label is first 8 characters of first transaction
- Location records the storage system the file is stored on (more than 1 possible)
- File paths have to be unique within a holding
- This is very different to previous system, it allows multiple versions of the file to be stored, as long as each version is in a different holding

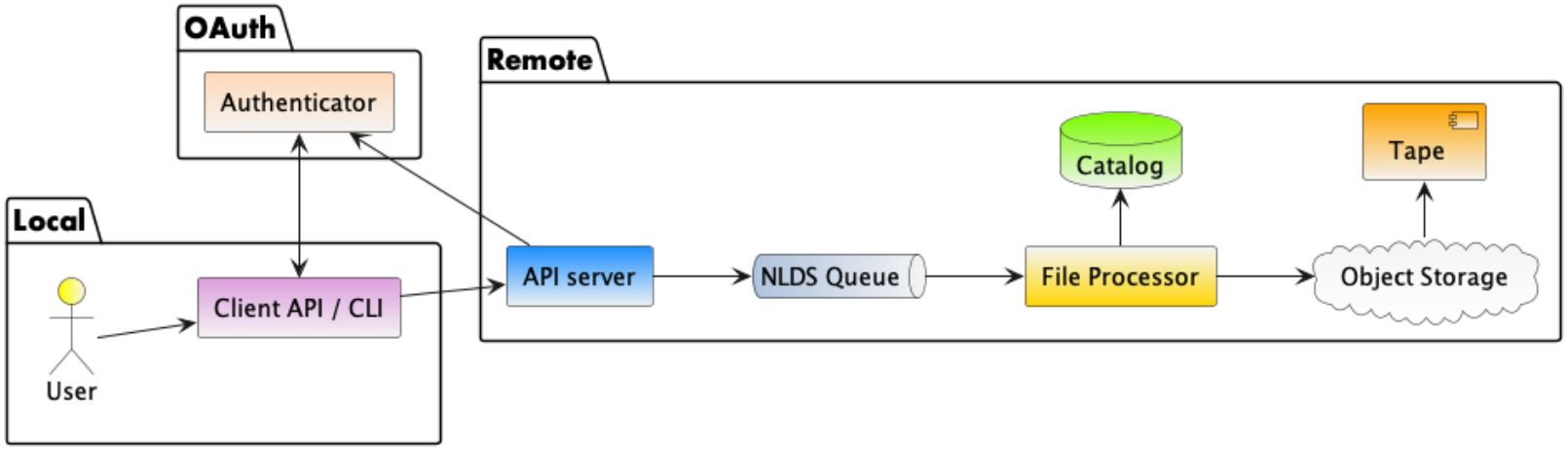


Architecture Advantages

- Security: all user interactions are authorised by OAuth2, using their JASMIN credentials
- Cloud available data: Object store allows data to be read across HTTP(S)
- Data is automatically catalogued on ingest
- Data can be tagged, searched by tags and regular expressions on paths
- The user does not interact with tape – they simply PUT or GET files
- Backup is handled automatically
- Highly fault tolerant – failed transfers will restart due to message broker

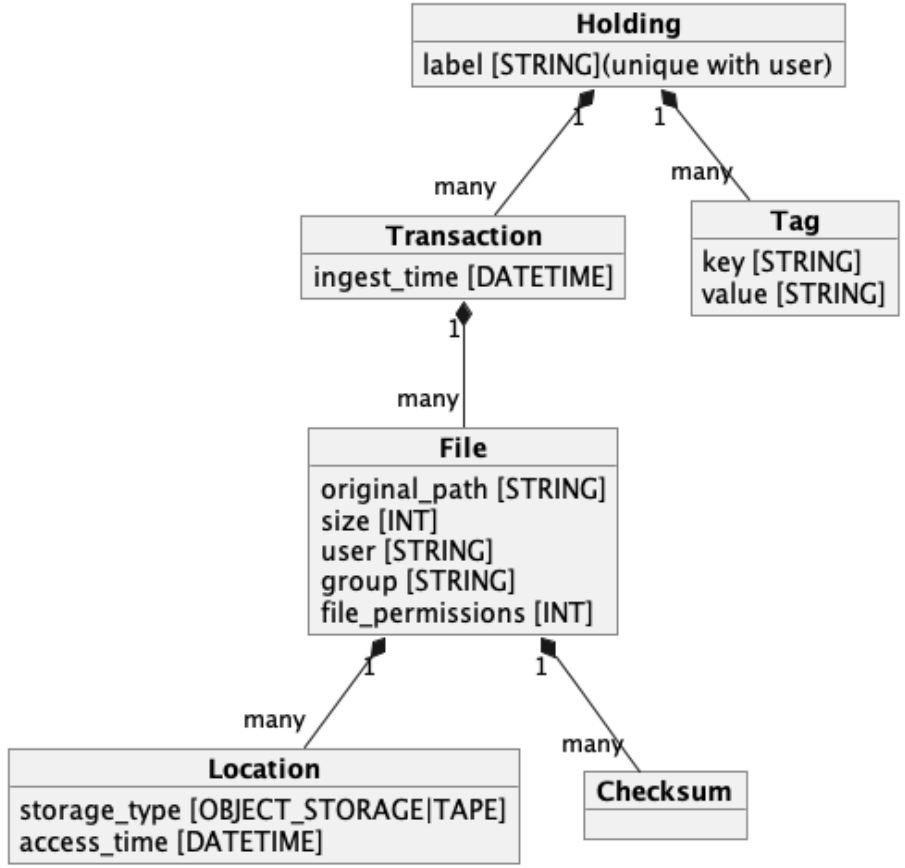
NLDS - User view

- Users interact with NLDS via a command line client or a Python client API
- Issue commands to copy data from the Disk (SOF, PFS or Pure) to the NLDS
- NLDS microservices copy the data on behalf of the user
- The data is cataloged
- Users can then delete the data from the disk and retrieve it from NLDS later



NLDS - User catalog

- **Holdings** are collections of files, that the user has chosen to collect together and assign a label to the collection
- **Transactions** record the user's action when PUTting a file into the NLDS. Each **holding** can contain numerous **transactions.**
- **Tags** can be associated with a **holding**, in a **key:value** format
- **File** records the details of a single **file**
- **Location** records the location(s) of a file, either on **OBJECT_STORAGE**, **TAPE**, both or neither



NLDS Client - User interaction

A command line client - `nlds` - with a small number of commands

- `put` and `putlist` – put a single file or a list of files to the NLDS
- `get` and `getlist` – get a single file or a list of files from the NLDS
- `meta` – add label and tags to a holding
- `stat` – determine the status of a transaction
- `list` – list a user's holdings, searching via label or tag with regular expressions supported
- `find` – list a user's files, searching via label or tag (for the holding containing the files) and filepath via regular expression

- https://cedadev.github.io/nlds-client/command_ref.html

NLDS Client - Commands with metadata

PUT a single / list of files

- `put --user --group (--label —-holding_id —-tag --job_label) filepath`
- `putlist --user --group (--label —-holding_id —-tag --job_label) filelist`
- Optional
  - `--label=`: if a holding with label exists then add to an existing holding with the label, otherwise create a new holding with the label
  - `--holding_id=`: adds to an existing holding with the (integer) id
  - `--tag=key:value`: adds a tag to the holding on PUT
  - `--job_label=`: set a label for the ingest job

GET a single / list of files

- `get --user --group --target (--label —-holding_id —-tag --job_label) filepath`
- `getlist --user --group --target (--label —-holding_id —-tag) filelist`

NLDS Client - Query with metadata

List the holdings for a user / group

- `list`
- Required arguments
  - `--user=`
- Optional arguments
  - `--group=`
  - `--holding_id=` (integer)
  - `--tag=key:value` (filter by tag)
  - `--label=` (filter by label)
  - `--time=datetime|(start datetime, end datetime)` (time the files were ingested)
  - `--regex` (use regular expressions when searching by label)
  - `--transaction_id=` (search by transaction id)

NLDS Client - Query with metadata

List / find the files for a user / group

- `find`
- Required arguments
  - `--user=`
- Optional arguments
  - `--group=`
  - `--holding_id=` (integer)
  - `--tag=key:value` (filter by tag for the holding containing the files)
  - `--label=` (filter by the holding label)
  - `--time=datetime|(start datetime, end datetime)` (time the files were ingested)
  - `--path=` (filter by original path, can be a substring, regex or wildcard)
  - `--regex` (use regular expressions when searching by label or path)
  - `--transaction_id=`(search by transaction id)
  - `--simple` (simple output of one file per line)
  - `--url` (output the object store URL)

NLDS Client - Query with metadata

Update the holding metadata

- `meta`
- Required arguments
  - `--user=`
  - One of (must guarantee uniqueness)
    - `--holding_id=`
    - `--label=`
  - Optional
    - `--group`
    - `--new_tag=key:value` (create or amend a tag)
    - `--del_tag=` (delete a tag)
    - `--new_label=` (change the label)

NLDS Client – Config file

- Quite a complex setup procedure
- Simplified by the `nlds init` command, which creates most of the config file
- Need to edit this to add user, group and object store credentials
- A comprehensive step-by-step guide has been written: https://cedadev.github.io/nlds-client/step_by_step.html
- There is also a detailed tutorial: https://cedadev.github.io/nlds-client/tutorial.html
- Installation of the client will be via PyPi (`pip install nlds-client`)

Current status

- Functionally complete, for `get/getlist`, `put/putlist` and the query commands
- In the final stages of the beta test
- Contact the JASMIN help desk if you would like to participate in this or future beta tests
- Almost ready for wider release – just waiting for production tape config to be finalised
- Will be fully available in the next two weeks
- … it's been a long time coming, so thanks for your patience

Future developments

- Iterative version releases through 2025->2026
- Non-breaking changes only
- Adding functionality such as `delete` and `cancel`, `quota` and `roles`
- User feedback is welcome, as it could shape future features
- Widen use to JASMIN users without GWS
- Moving ET and JDMA batches / migrations to NLDS
- This will be done GWS by GWS
- Does not require moving data, just copying and transforming the records to the NLDS database
- ET and JDMA will be retired when transfer of assets to NLDS is complete

Summary

- NLDS is a hierarchical file management system which is:
  - Portable
  - CRUD
  - Single interface for hot/warm/cold storage
- Comprised of client, api-server and microservices
- Will be the sole method of interacting with tape for JASMIN GWS users in the future

UKRI Science and Technology Facilities Council | Natural Environment Research Council

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Thank you!

Any Questions?

neil.massey@stfc.ac.uk

https://github.com/cedadev/nlds

https://github.com/cedadev/nlds-client

NLDS was supported through the ESiWACE2 project. The project ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988.

Centre for Environmental Data Analysis | JASMIN | Scientific Computing | National Centre for Atmospheric Science | National Centre for Earth Observation | RAL Space