ICAT

Data Server (IDS)

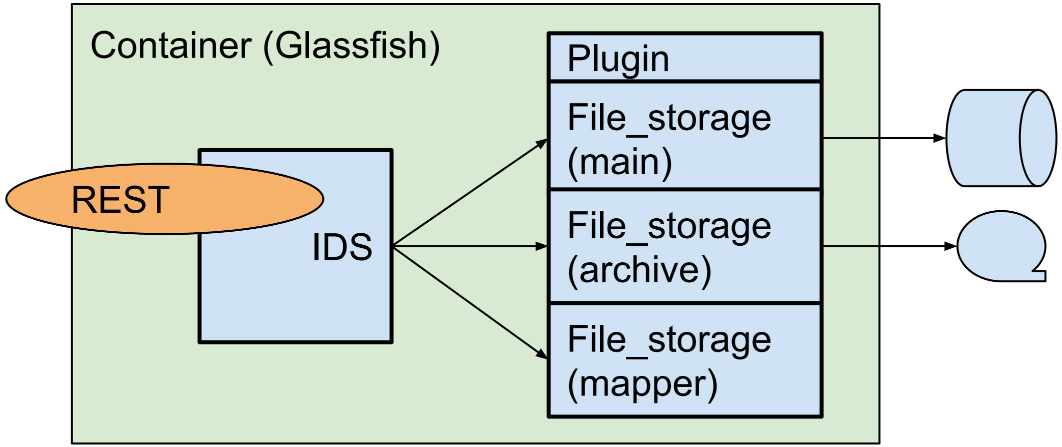
Steve Fisher <dr.s.m.fisher@gmail.com>

Outline

- What it is
- Configuring the IDS
- Writing a plugin
- Security Issues

IDS Server

- Can use two level storage if not practical to keep all data on low latency storage
  - IDS manages movement between main and archive storage. Calls to read data not in main storage triggers restore and returns failure.
  - Can use explicit archive and restore calls
- Authz linked to ICAT



IDS Server - calls

- Storing a file
  - Send in the id of the dataset and name of file
  - Look up dataset in ICAT and check that can write
  - Use the plugin to write the file - to main storage
  - Catalog the datafile in ICAT
  - Return the id of the new datafile
- Reading one file
  - Check with ICAT that file is readable
  - If file in main storage plugin streams data directly to the user
- Reading multiple files
  - Check with ICAT that all files are readable
  - The same mechanism but a zip file is created on the fly

Ids.properties - 1

- icat.url: Normally just the scheme, the hostname and the port. https://example.com:443
- plugin.zipMapper.class: The class name of the ZipMapper which defines the Zip file structure you want. The class must be deployed in the lib/applibs directory of your domain and must be packaged with all its dependencies. org.icatproject.ids.storage.ZipMapper
- plugin.main.class: The class name of the main storage plugin. The class must be deployed in the lib/applibs directory of your domain and must be packaged with all its dependencies. org.icatproject.ids.storage.MainFileStorage
- plugin.main.properties: Optional property file for the main storage plugin. ids.storage_file.main.properties
- cache.dir: The location (absolute or relative to the config directory of the domain) of a directory to hold mostly zip files. ../data/ids/cache
- preparedCount: The number of preparedId values from prepareData calls to remember. 10000
- processQueueIntervalSeconds: The frequency of checking the process queue. This is used both for cleaning old information from memory and for triggering movements between main and archive storage (if selected). 60
- sizeCheckIntervalSeconds: How frequently to check the cache sizes and clean up if necessary. 60

Ids.properties - 2

- readOnly: If true disables write operations (put and delete).
- linkLifetimeSeconds: The length of time in seconds to keep the links established by the getLink call. If this is set to zero then the getLink call will return a NotImplementedException. 3600
- reader: Space separated icat plugin name and credentials for a user permitted to read all datasets, datafiles, investigations and facilities. db username root password secret
- [key: Contributes to the computation of a cryptographic hash added to the location value in the database - see later]
- maxIdsInQuery: The number of literal id values to be generated in an ICAT query. For Oracle this must not exceed 1000. 1000
- [Log.list: Set of call types to log via JMS calls. The types are specified by a space separated list of values taken from READ, WRITE, LINK, MIGRATE, PREPARE and INFO. READ WRITE INFO LINK MIGRATE PREPARE]
- [Logback.xml: Path to a logback.xml file.]
- rootUserNames: A space separated list of users allowed to make the getServiceStatus call. The user name must include the mechanism if the authenticators have been configured that way. root

Ids.properties - archive storage

Omit all these to only have one level storage

- plugin.archive.class: The class name of the archive storage plugin. org.icatproject.ids.storage.ArchiveFileStorage
- plugin.archive.properties: Optional property file for the archive storage plugin. Ids.storage_file.archive.properties
- storageUnit: May be dataset or datafile Dataset
- writeDelaySeconds: The amount of time to wait before writing to archive storage. This exists to allow enough time for all the datafiles to be added to a dataset before it is zipped and written. 60
- startArchivingLevel1024bytes: If the space used in main storage exceeds this then datasets will be archived (oldest first) until the space used is below stopArchivingLevel1024bytes. Set to around 90% of available man storage space
- stopArchivingLevel1024bytes: Set to around 85% of available space
- tidyBlockSize: The number of datafiles or datasets to get back in one call for archive request when space on main storage is low. 500

Ids.properties - file checking

- filesCheck.parallelCount: This must always be set, and if non zero then the readability of the data will be checked. The behaviour is dependent upon whether or not archive storage has a been requested.
- filesCheck.gapSeconds: the number of seconds to wait before launching a check of the next batch of datafiles or datasets.
- filesCheck.lastIdFile: the location of a file which is used to store the id value of the last datafile or dataset to be checked. This is so that if the IDS is restarted it will continue checking where it left off. ../data/ids/lastIdFile
- filesCheck.errorLog: the file with a list of errors found. ../data/ids/errorLog

Writing a plugin

- The plugin can store the data however it chooses.
- Maybe constrained by large volume of existing data.
- Read: https://repo.icatproject.org/site/ids/plugin/1.3.1/manual.html
- And look at the ids.storage_file as an example https://github.com/icatproject/ids.storage_file/blob/master/src/main/java/org/icatproject/ids/storage/MainFileStorage.java

Security considerations

Again read: https://repo.icatproject.org/site/ids/plugin/1.3.1/manual.html

Your plugin constructs the location field when storing a file.

Main problem: generating a Datafile (via ICAT) somewhere with an existing location field accessing other people’s data.

Deletion is not a problem since ids 1.7.0 but reading is.

Safe simple solution - cryptographic hash

- Generate hash of
    a. "id" of the datafile
    b. "location" value as seen by the plugin
    c. key known only to the IDS and which is defined in the ids.properties file.
- The value stored in icat is the location followed by the hash separated by a space. The hash value goes at the end to help with indexing.
- “Impossible” to generate a Datafile.location field that the ids will accept without knowing the secret IDS key
- When plugin needs location
    a. Start with Datafile.location
    b. Check that the hash has the expected value
    c. Pass the first part of the location field is to the plugin
- To support the new clone functionality of the icat.server the icat.properties must have the same key value to compute the correct location value for the Datafile clone.