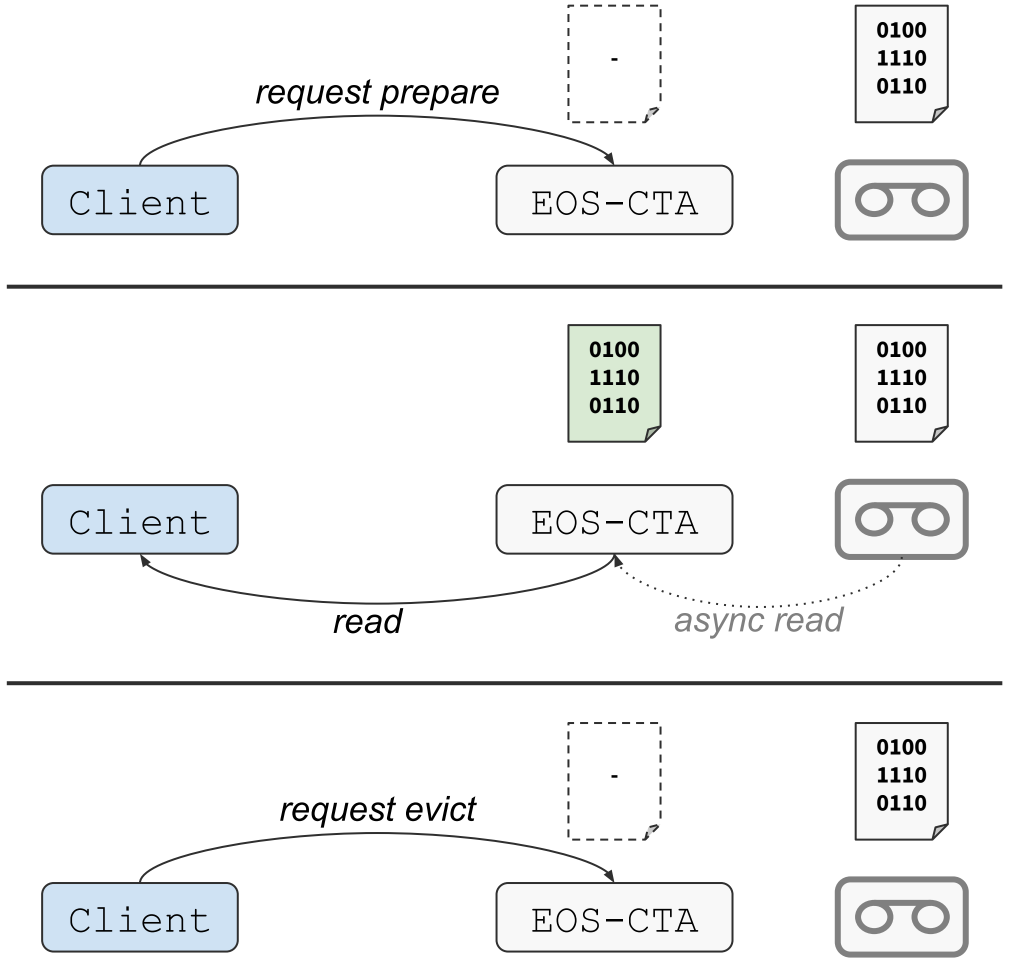
From *stagerrm* to *evict* : Disk buffer management in CTA

Presented by:
João Afonso
CERN, IT-SD-TAB

Eviction on EOS-CTA

- **Shift in paradigm:**
  - CTA workflow requires **explicit eviction** of disk replicas, instead of relying on **garbage collection**.
- **Advantages:**
  - Requires a smaller disk buffer
  - Favours higher throughput
  - Clients have control over each disk copy lifecycle:
    - Works very well with CERNs File Transfer Service (FTS)!
- Alternative methods (i.e. garbage collection) are still used, but only as a fallback option



Eviction on EOS-CTA

However…

- Reused old CASTOR command – **stagerrm** – misaligned from EOS-CTA workflow language:
  - PREPARE
  - EVICT_PREPARE

```
$ eos root://eoscta stagerrm /path/to/file
```

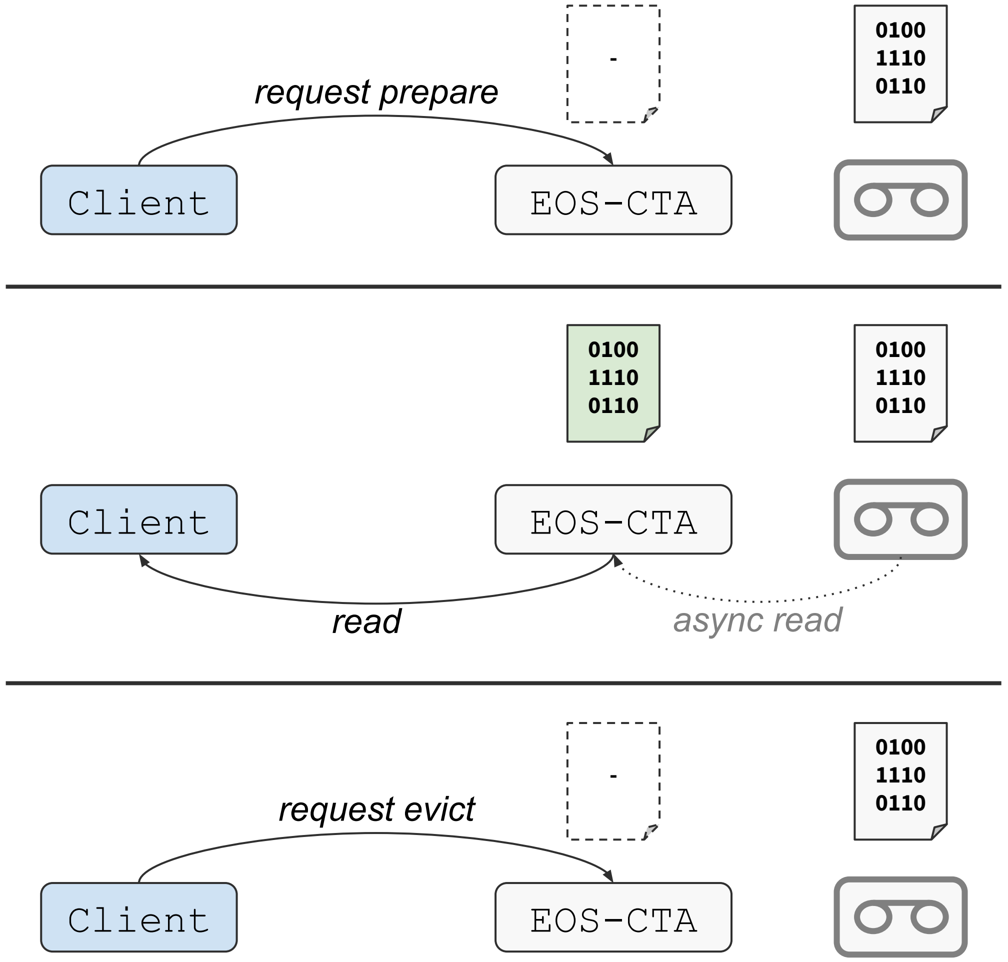
Old language

```
$ xrdfs root://eoscta prepare -e /path/to/file
```

Evict - New language

- And showed some unresolved problems…

Note:
- **stagerrm/evict** is a similar in concept to:
  - **Unpin**, on dCache
  - **Release**, on the HTTP Tape REST API

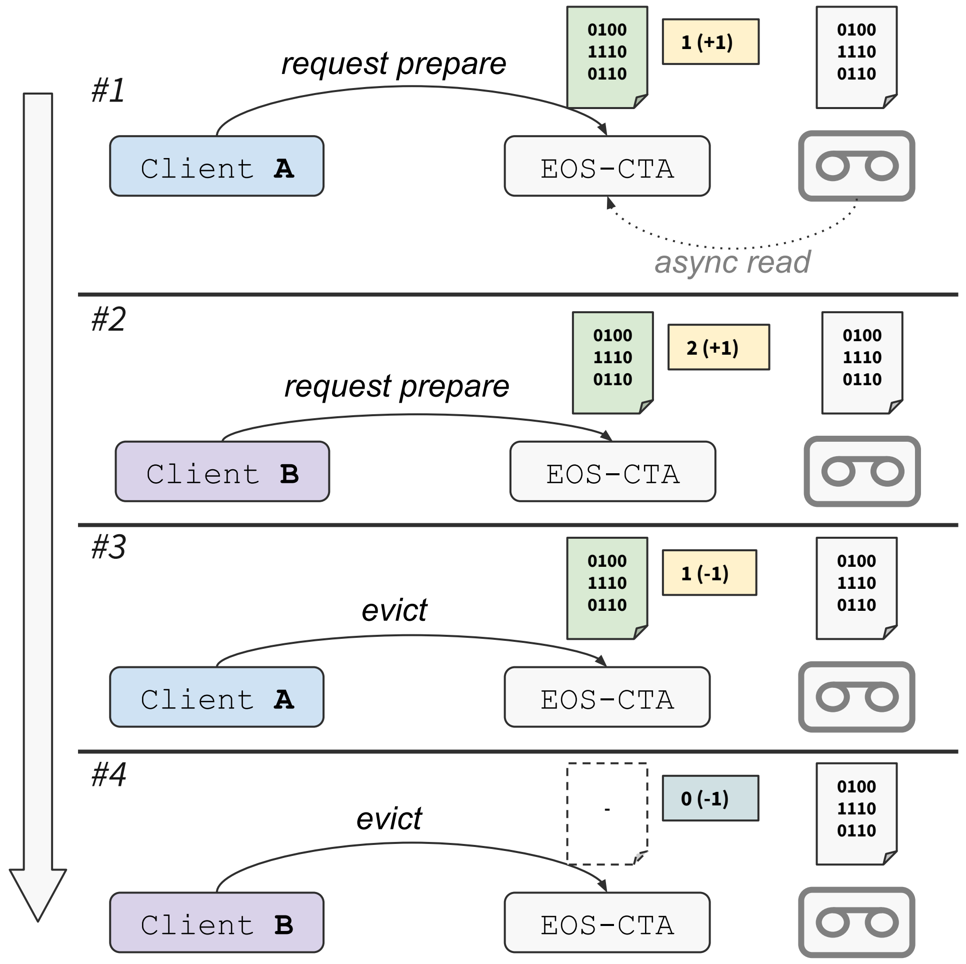


CERN Tape Archive

Problem 1: Handling multiple requests per file

Partially fixed:

- Every file retrieved to disk contains an **evict counter**:
    - Keeps track of number of requests.
    - File is removed only after all clients have evicted.
    - Value stored as EOS file metadata.



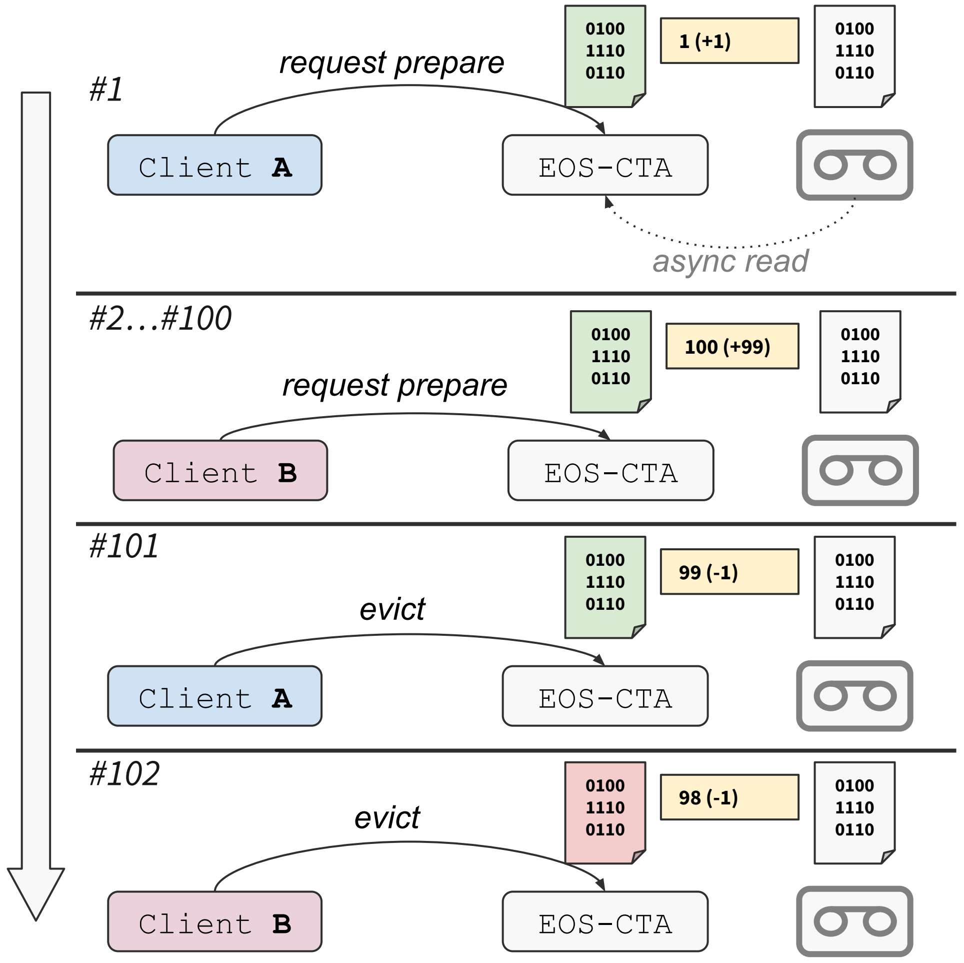
Problem 1: Handling multiple requests per file

Partially fixed:

- Every file retrieved to disk contains an **evict counter**:
  - Keeps track of number of requests.
  - File is removed only after all clients have evicted.
  - Value stored as EOS file metadata.

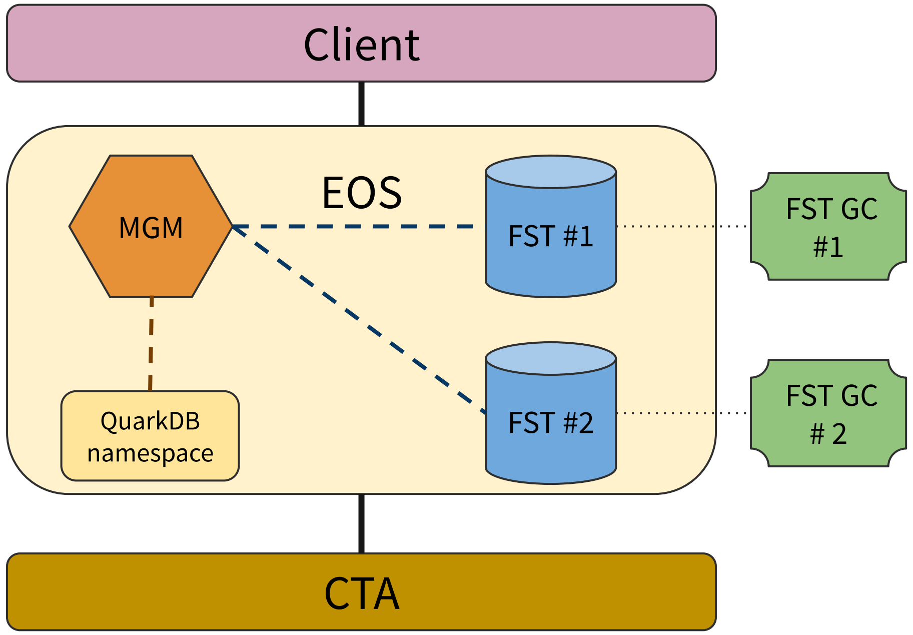
Evict counter could be abused…

- Bad clients could request the same files 1000's of times, without evicting each time.
- Fallback garbage collector could not cope:
  - Internally, it called **stagerrm** once per iteration, which could only reduced the evict counter by 1.
  - As a result, disk copies would get stuck on disk.



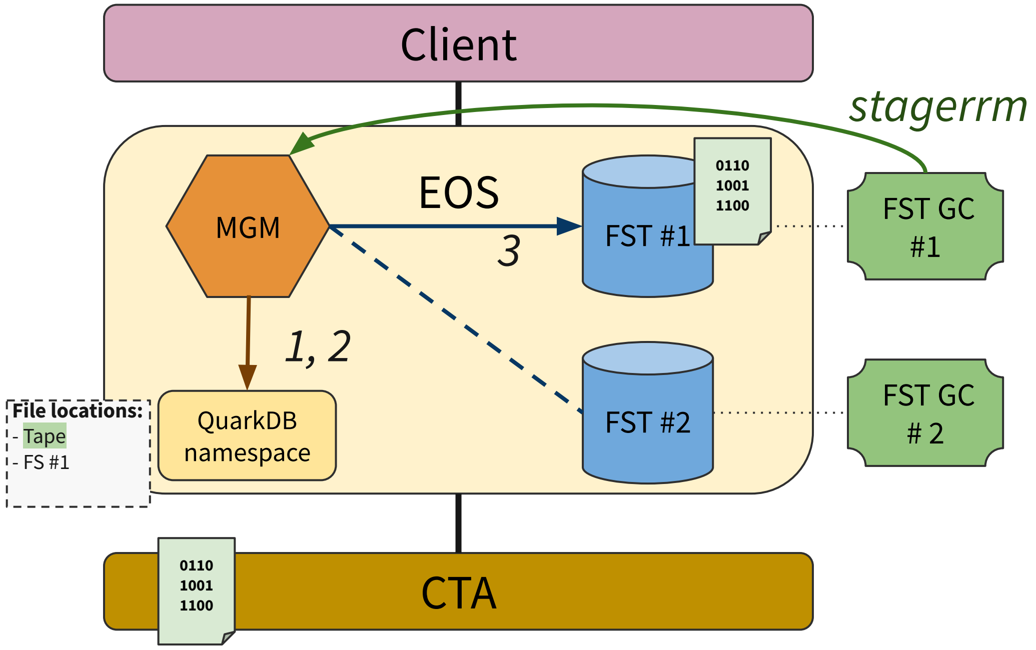
Problem 2: Handling multiple FS disk replicas

- Each FST on a EOS-CTA instance is backed by a **FST Garbage Collector**:
  - Clears old disk replicas when they were not properly evicted.
  - Guarantees that each FST always has some free space available → <u>Required for maximum throughput.</u>
- The **FST Garbage Collector (GC)** uses the **stagerrm** command that it wants a file to be removed.



Problem 2: Handling multiple FS disk replicas

- Each FST on a EOS-CTA instance is backed by a **FST Garbage Collector**:
  - Clears old disk replicas when they were not properly evicted.
  - Guarantees that each FST always has some free space available → Required for maximum throughput.
- The **FST Garbage Collector (GC)** uses the **stagerrm** command that it wants a file to be removed.

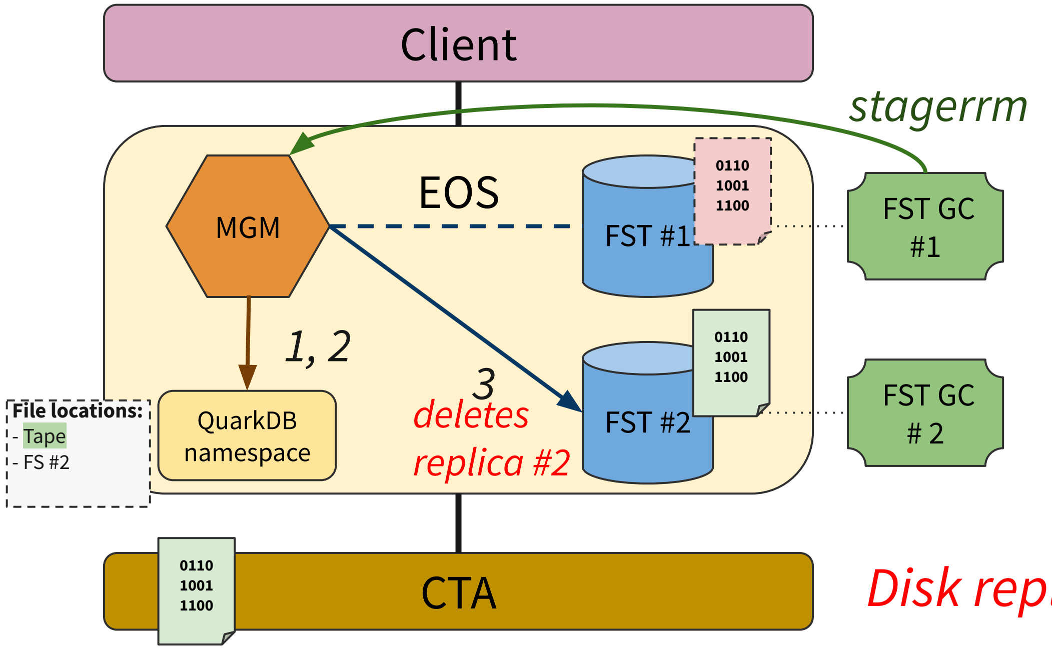


When the EOS **MGM** receives a **stagerrm** command, it does:

1. Confirm that a tape replica exists.
2. Find all disk replica locations.
3. Clear all disk replicas from the FSTs.

Problem 2: Handling multiple FS disk replicas

- Each FST on a EOS-CTA instance is backed by a **FST Garbage Collector**:
  - Clears old disk replicas when they were not properly evicted.
  - Guarantees that each FST always has some free space available → <u>Required for maximum throughput.</u>
- The **FST Garbage Collector (GC)** uses the **stagerrm** command that it wants a file to be removed.



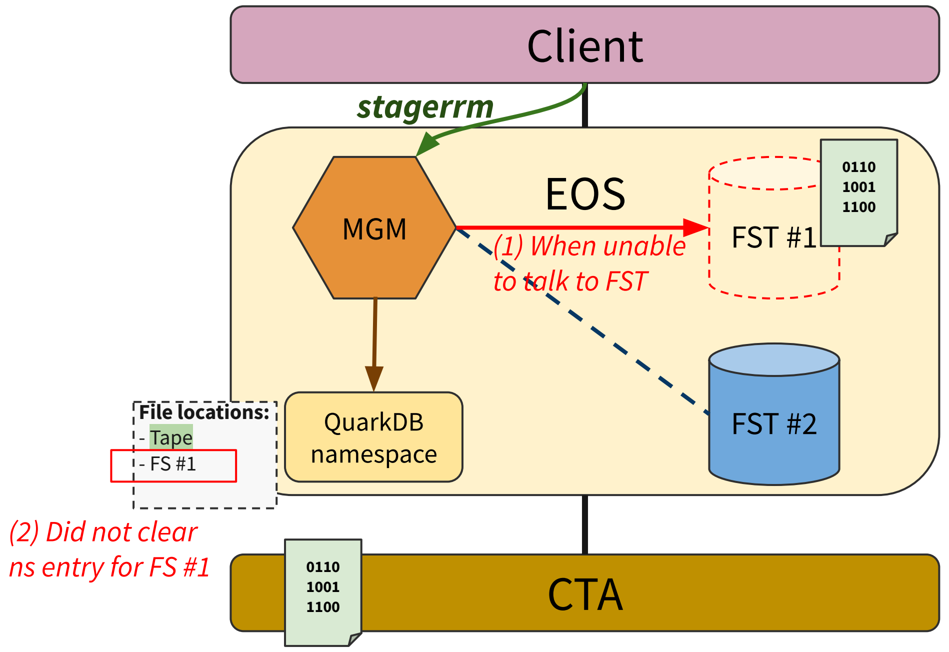
However, this setup can cause conflict between different FSTs. For example, in the following scenario:

a. Garbage collector #1 keeps trying to delete stub file on FST #1, but is unable to since file no longer on EOS namespace (data got corrupted).
b. As soon as a new replica is staged on FST #2, it will be deleted by the MGM.
c. Client is unable to get the file!

Disk replica #2 is wrongly deleted, by GC #1, instead of replica #1

Problem 3: Removing files from unresponsive FSTs/disks

- If the MGM was not able to contact the FST during a **stagerrm**, then it would never remove the corresponding disk replica metadata from the EOS namespace (after a failed FS drain).
- This would keep showing the file on disk – even if the disk was broken – which would result in new disk replicas not getting created.
- The command “*eos file drop <fsid>*” exists for dropping FS entries, but is not safe:
  - We needed a command that always protected Tape replica metadata.



Replacing stagerrm with evict

- Solving these problems required several changes to the EOS **stagerrm** command.

$$\Downarrow$$

- Opportunity to create new EOS command and ditch the CASTOR old terminology!

Replacing stagerrm with evict

- Solving these problems required several changes to the EOS **stagerrm** command.

↓

- Opportunity to create new EOS command and ditch the CASTOR old terminology!

Solution:

- New EOS **evict** command:
  - Adapted to EOS-CTA use case and language.
  - Exists side-by-side with the old **stagerrm** command, which will be deprecated.
    - Allows for a smooth transition for operator tools.
  - Provides solutions to the previously shown problems.

eos evict - New command

```
$ eos evict
Usage: evict [--fsid <fsid>] [--ignore-removal-on-fst] [--ignore-evict-counter] <path>|fid:<fid-dec>]|fxid:<fid-hex>
[<path>|fid:<fid-dec>]|fxid:<fid-hex>] ...
    Removes disk replicas of the given files, separated by space

  Optional arguments:
    --ignore-evict-counter  : Force eviction by bypassing evict counter
    --fsid <fsid>           : Evict disk copy only from a single fsid
    --ignore-removal-on-fst : Ignore file removal on fst, namespace-only operation

    This command requires 'write' and 'p' acl flag permission
```

- This a mirror of the `xrdfs prepare -e` command, with extra features for operators.
- Code is reused between both (references to *stagerrm* removed from code).

```
$ xrdfs root://eoscta prepare -e /path/to/file
```

Better interaction with the evict counter

```
$ eos evict /path/to/file
```

- Simplest use of EOS **evict**.
- Functionally the same as **stagerrm**:
  - The evict counter is reduced by 1 on every call.
  - Once evict counter reaches zero, all disk replicas will be deleted.
- Exactly the same behaviour as:

  -
    ```
    $ xrdfs root://eoscta prepare -e /path/to/file
    ```

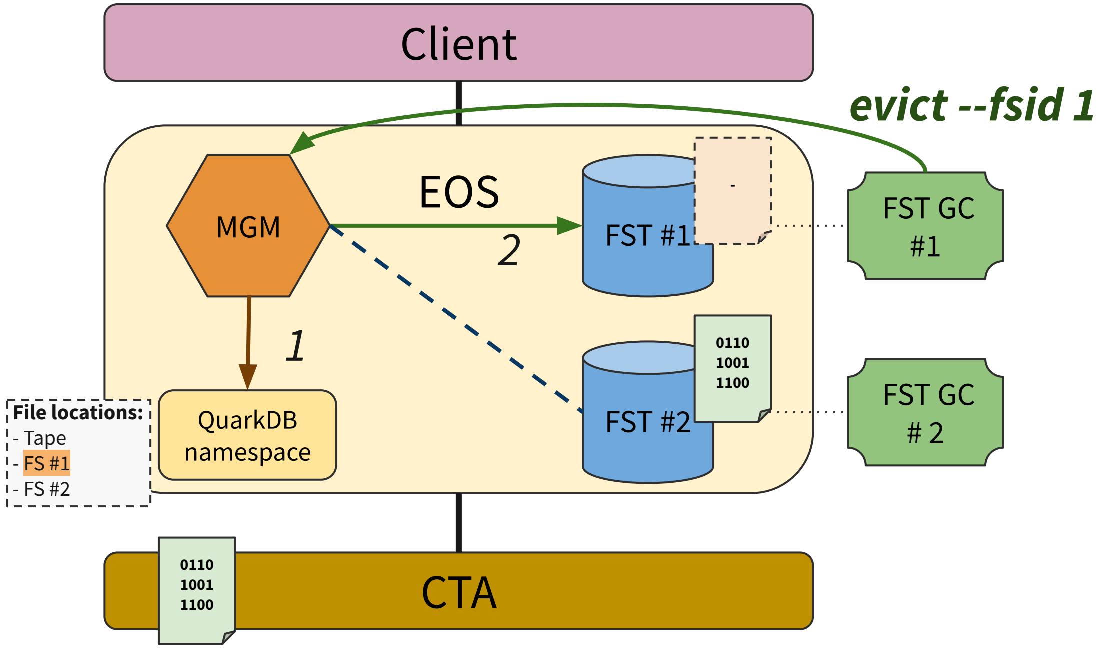
```
$ eos evict /path/to/file --ignore-evict-counter
```

- Extra option to bypass the *evict counter*.
- Forces the *evict counter* to zero:
  - This will trigger the immediate removal of all disk replicas.

Option to delete only a single disk copy

```
$ eos evict /path/to/file --ignore-evict-counter --fsid 123
```

- Will trigger only the removal of the disk replica with the fsid passed as argument.
- Other disk replicas (and *evict counter* value) are preserved.
- For consistency, the argument "*--ignore-evict-counter*" is mandatory with "*--fsid <id>*".
- To be used by tools that work on the individual FST level, such as the FST garbage collector.



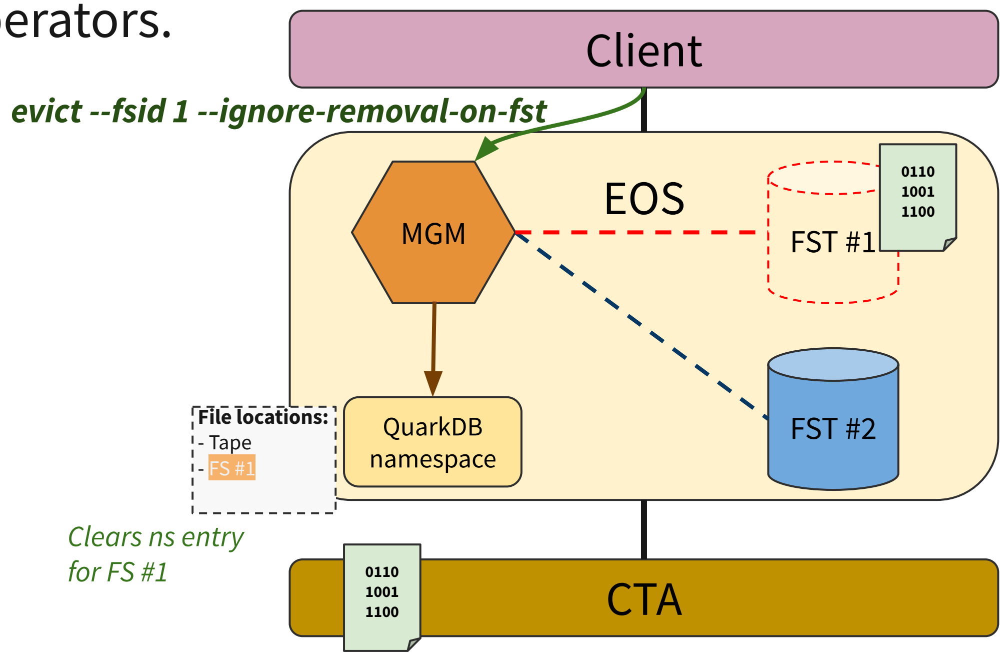
Example:

- The FST garbage collector #1 can use `*--fsid 1*` to trigger the removal of only the the file replica on FST #1.
- The file replica on FST #2 will be preserved.

Option to delete only the EOS namespace

```
$ eos evict /path/to/file --ignore-evict-counter --fsid 123 --ignore-removal-on-fst
```

- Will clear the fsid entry on the EOS namespace, without checking the FST.
- Useful when the FST/disk is dead and unreachable.
- For consistency, "*--ignore-evict-counter*" and "*--fsid <id>*" are mandatory.
- Safe alternative to "*eos file drop <fsid>*".
- To be used by tools or operators.



- File no longer reachable, but this is OK on a dead disk buffer.

Current state

- New command EOS **evict**, and all its features, available from EOS **version 5.2.4**.

- Updated FST garbage collector logic included on CTA version **4/5.10.9.0-1**.
    - Available on **next public release**.

home.cern