

1 Queuing an archive request in CTA

Figure 1 is a sequence diagram showing how an EOS instance, the CTA front-end, scheduler, catalogue and object-store interact in order to queue an archive request. The sequence diagram has the following three prerequisites:

- A CTA administrator has entered the archive route(s) for the storage class of the file to be archived.
- A CTA administrator has entered the archive mount policy for the owner of the file.
- A file has successfully been written to an EOS instance and the workflow engine of that instance is about to queue a request to archive the file in CTA.

The front-end is responsible for receiving and authorising requests from CTA administrators and EOS instances and passing them on to the Scheduler and Catalogue objects. Step 1 of the figure 1 shows the EOS instance initiating the scenario by requesting the CTA front-end to queue an archive request. The CTA front-end is an XRootD plugin and relies on the underlying xrootd daemon to authenticate the EOS instance. Step 2 shows the front-end plugin authorising the EOS instance. Step 3 shows the front end passing on the now authorised queue request to the scheduler. The scheduler interacts with two persistent stores, the catalogue which is responsible for storing all of the critical metadata of the CTA system and the object-store (OStoreDB) which is responsible persisting the archive and retrieve request queues of CTA. The catalogue and OStoreDB do not communicate directly with each other, the scheduler is responsible for pulling information from one store and pushing it into the other. Step 4 shows the scheduler calling the `prepareForNewFile()` method of the catalogue in order to get all of the information required to queue the request. The `prepareForNewFile()` method of the catalogue has to provide 4 pieces of information. It has to generate and provide a unique archive identifier for the file, it has to use the storage class of the file to lookup and provide the destination tape pool(s), it has to use the owner of the file to lookup and provide the owner's archive mount policy and finally it has to serialise all of the remaining metadata of the file into a blob that can be used once the file has actually been written to tape in order to record the full result of the archived file transfer into the catalogue. The catalogue only records the end result of files being written to tape. The catalogue does not contain any information about individual files that are "on their way" to tape. The blob will be kept within the OStoreDB until the moment when the file has been written to tape, at which point the blob will be given back to the catalogue for permanent storage.

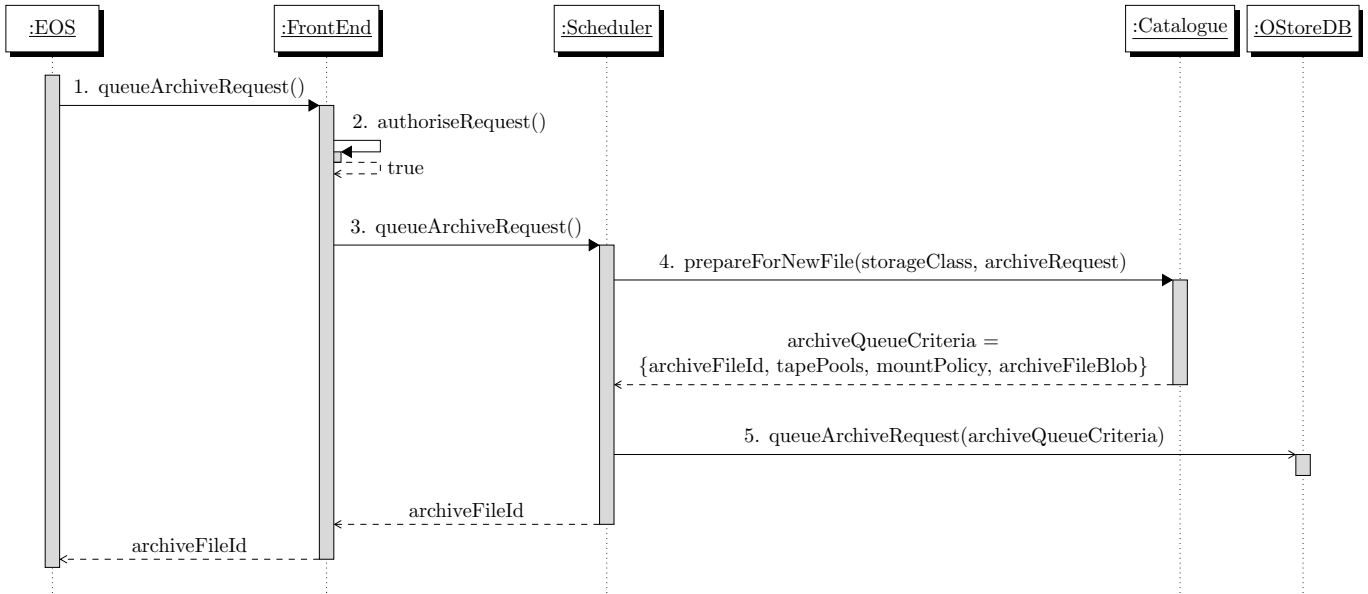


Figure 1: Queuing an archive request

2 Queuing a retrieve request in CTA

Figure 2 is a sequence diagram showing how an EOS instance, the CTA front-end, scheduler, catalogue and object store interact in order to queue a retrieve request. The sequence diagram has the following three prerequisites:

- A file has been written to EOS, archived to tape and then had just its EOS disk copy deleted. The file is still referenced in the EOS namespace and has one or more copies safely stored on tape.
- A CTA administrator has entered a recall mount policy for a user who wishes to retrieve the file back from tape.
- The EOS user wanting the file has just requested EOS to retrieve it from tape.

Step 1 of figure 2 shows EOS initiating the scenario by requesting the CTA front-end to queue a request to retrieve the file. In steps 2 and 3 the front-end authorises the request and passes it on to the scheduler. At step 4 the scheduler calls the `prepareToRetrieveFile()` method on the catalogue. The catalogue must return everything it knows about the file that is pertinent to its retrieval. The `prepareToRetrieveFile()` method must therefore return the locations of all of the file's tape copies and the policy to be used to decide when to mount a tape for retrieval. If there is more than one copy on tape, then it will be the responsibility of the scheduling algorithm to decide which one is finally passed on to the tape servers. The location of a tape copy shall include the following 4 pieces of information:

- The volume identifier of the tape.
- The file sequence number of the tape file.
- The block ID of the tape file.

A mount policy contains many criteria; an example of such a criteria is the minimum amount of data required to warrant a tape mount. The catalogue does not communicate with the object store (OStoreDB) directly and therefore at step 5 the scheduler passes the result of `prepareToRetrieveFile()` onto the OStoreDB.

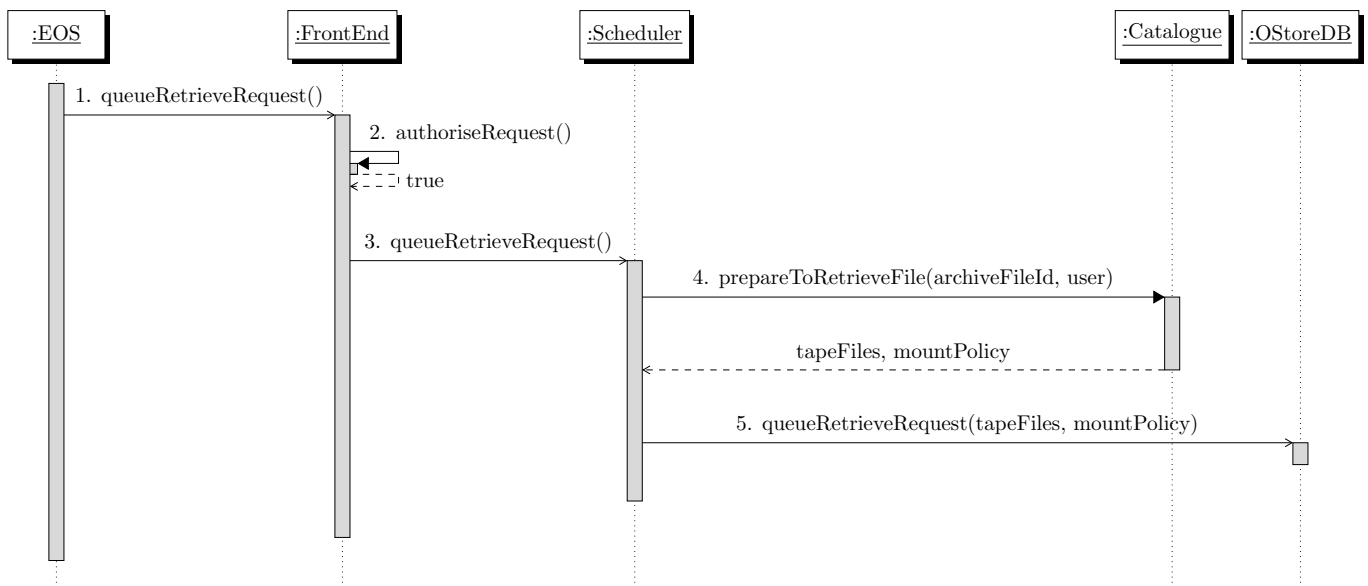


Figure 2: Queuing a retrieve request