EPJ Web of Conferences **295**, 01008 (2024) https://doi.org/10.1051/epjconf/202429501008
CHEP 2023

An HTTP REST API for Tape-backed Storage

João Afonso[1,*], *Cédric* Caffy[1], *Mihai* Patrascoiu[1], *Julien* Leduc[1], *Michael* Davis[1], *Steven* Murray[1], and *Pablo* Cortes[1]

[1]CERN, Esplanade des Particules 1, 1211 Geneva 23, Switzerland

Abstract. The goal of the HTTP REST API for Tape project is to provide a simple, minimalistic and uniform interface to manage data transfers between Storage Endpoints (SEs) where the source file is on tape. The project is a collaboration between the developers of WLCG storage systems (EOS+CTA, dCache, StoRM) and data transfer clients (gfal2, FTS).

For some years, HTTP has been growing in popularity as the preferred data transfer protocol between many WLCG SEs. However — unlike other protocols such as XRootD and SRM — HTTP does not include a method to stage files from tape to disk prior to transfer, forcing the use of workarounds such as hybrid protocols (different protocols used for the "stage" and "transfer" parts of the operation). The HTTP REST API offers a simple and consistent solution, by extending the HTTP protocol to include staging operations. It provides clients with a common and consistent API across different storage systems to manage and monitor the disk and tape residency of stored files.

In this contribution, we present the history and motivation of the HTTP REST API project, the specification of version 1 of the API and implementation details in the various storage and data transfer systems. We also describe our experience of its deployment and use for LHC Run-3 operations. We conclude with a discussion of possible future work.

1 Introduction

The HTTP protocol is being used more and more within the Worldwide LHC Computing Grid (WLCG) [1] for data transfer between different Storage Endpoints (SEs). Unfortunately, unlike the Storage Resource Manager (SRM) [2] and XRootD protocols [3], HTTP does not provide a mechanism to handle tape-related metadata operations. Amongst the operations not supported by HTTP we have: tape file disk staging (copying a file from tape into a disk buffer, so that it can be directly read by the client—this is also known as *bringOnline* in SRM or *prepare* in XRootD); tape archival tracking (checking if a file resides on tape and/or disk); and tape-stored file disk eviction (delete the disk/buffer copy of a file that is already on tape).

The absence of these operations in HTTP forced users to put in place complex workarounds. For example, by using HTTP—to perform data transfers between disk instances—in conjunction to a different protocol (SRM or XRootD)—to perform tape-related operations [4].

*e-mail: joao.afonso@cern.ch

Therefore, the main motivation for the WLCG Tape REST API was to simplify end-users' life, by providing tape-related metadata operations via HTTP—in addition to the already existing data transfer functionality.

In 2022, the WLCG Tape REST API work group concluded its efforts on a common HTTP REST interface that allows clients to manage access to files stored on tape. This collaboration between the developers of the various WLCG storage systems (e.g. EOS+CTA [5–7], dCache [8, 9], StoRM [10, 11]) and data transfer clients (e.g. gfal2, FTS [12, 13]) resulted in a new protocol to be supported by all these storage archival services, finally allowing all tape file transfer and management operations to be done with HTTP (the Tape REST API reference document can be found on [14]). With this, both the SRM or XRootD protocols are no longer a requirement for tape-related operations in WLCG.

At CERN, the HTTP Tape REST API has been added to EOS+CTA, thus allowing all Tier-0 storage-on-tape operations to be controlled via HTTP. Its production deployment was and is now being used for LHC Run-3 operations.

With this paper we will explain the motivation and use cases that lead to the specification of the HTTP Tape REST API, followed by the details of how it was implemented in EOS+CTA. We will then proceed with discussing our experience with deploying it for Run-3 operations and finally conclude with an overview of future work.

1.1 Previous work

The process that led to the creation of the HTTP Tape REST API started around 2018, with a movement in the WLCG community to migrate disk-to-disk transfers to HTTP-TPC (HTTP third-party copy). At the time, most WLCG Tier-1 transfers were using GridFTP, which had to be deprecated (more details about this can be found on [15]). While most of WLCG Tier-1 disk storage endpoints moved to HTTP-TPC early-on, the disk part of tape-endpoints had to remain unchanged for a while waiting for extra development work (due to HTTP not supporting tape operations).

Subsequently, a SRM + HTTP-TPC solution was put in place to replace SRM + GridFTP [4] in Tier-1 tape-endpoints. This process occurred in 2020/2021 and required site-issued tokens (most widely used implementation being *macaroons* [16]) plus support in FTS/Gfal2 and SEs [17]. Tape operations still had be done with SRM *bringOnline* [2].

Around 2022, CTA at CERN deployed direct HTTP access for the disk part of its Tier-0 tape-storage endpoints. Tape operations in CTA were still only possible with XRootD.

Finally, in 2021, discussions started to replace SRM tape operations with a Tape REST API. This would finally allow all transfers—both disk and tape—to be managed with HTTP. Both SRM and XRootD would no longer be required.

Initially, there was a proposal to extend the pre-existing dCache HTTP REST interface with bulk operations (dCache Bulk Requests API). This project was not followed up, but its details can be found on the WLCG DOMA meeting notes [18]. However, this served as a kickstarting point for the WLCG Tape REST API project.

For Tape REST API the project to be completed, there were several differences between WLCG Tape Storage Endpoints (SEs) that had to be discussed. In particular, the XRootD based SEs didn't work the same as SRM based SEs: for example, there is no support for disk file pinning in XRootD (stage the file from tape and keep its disk replica for several hours), while this is natively supported by SRM. Another difference is that dCache requests are per-bulk based, while EOS+CTA/StoRM/FTS are per-file based.

It was therefore necessary to start discussing and implementing a new common WLCG tape REST API interface that would allow to perform tape metadata operations regardless of the underlying storage technology used by SEs.

2 Use cases

The HTTP Tape REST API provides a common interface for managing tape file metadata operations purely with HTTP (the full specification—final output of the WLCG Tape REST API work group—can be found in the Tape REST API Reference Document [14]). These operations provide a mechanism for controlling and checking file residency on disk or tape. By using the API, the client has control over the archival or retrieval of files from tape.

On EOS+CTA, the new HTTP Tape REST API has a similar behaviour to the previously implemented XRootD protocol for tape file transfers. However, there are some significant differences.

One important change—relevant for the remaining part of this paper—is the concept of a Bulk Stage Request. On the HTTP Tape REST API, each stage request corresponds to a named resource (the Bulk Request) that persists on the server side. They can be created, read, modified or deleted using the corresponding HTTP methods. Since the list of files is stored with the Bulk Request, the server only needs its *Request ID* to know the files it contains.

The EOS+CTA implementation of XRootD, on the contrary, does not store any Bulk Request information: there is no object storing the list of files associated with each request. Instead, the *Request ID* is assigned to the attributes of each one of the requested files. As a consequence, it is not possible for the server to check the requested files based solely on the *Request ID* (it would need to scan the metadata of all existing files). Instead, to track the progress of one request with XRootD, the client needs to pass a list with the names of all files, every time.

On the remaining parts of this section we summarise how both protocols compare to each other on CTA+EOS.

2.1 Tracking file archival on tape

Both protocols can provide information about the progress of writing files to tape.

Use case	**XRootD**	**HTTP Tape REST API**
Check disk/tape residency of files being archived to tape	*xrdfs query prepare ⟨dummy_id⟩ /path/file1.txt /path/file2.txt ...*	*POST /api/v1/archiveinfo [with JSON containing list of files]*

In XRootD, the same command can be used to check both the progress of the archivals and stagings of files to/from tape. In addition to this, when tracking an archival there is no retrieve *Request ID* yet, which means we can set it as any dummy value.

In the HTTP Tape REST API, on the contrary, we just need to pass a json object containing the list of file paths.

For more details see also the specification document, section *ARCHIVEINFO* [14].

2.2 Staging a set of files from tape

Both protocols define mechanisms for managing the staging files from tape. A stage request orders the Storage Endpoint to make the requested files available with disk-like latency, usually by copying them to a disk buffer.

Use case	XRootD	HTTP Tape REST API
Submission of a stage request, containing multiple files; as a result, a ⟨*request_id*⟩ is generated	*xrdfs prepare -s /path/file1.txt /path/file2.txt ...*	*POST /api/v1/stage [with JSON containing list of files]*
Track the progress of a bulk-request	*xrdfs query prepare ⟨dummy_id⟩ /path/file1.txt /path/file2.txt ...*	*GET /api/v1/stage/⟨request_id⟩*
Cancel multiple files from a stage bulk-request	*xrdfs prepare -a ⟨request_id⟩ /path/file1.txt /path/file2.txt ...*	*POST /api/v1/stage/⟨request_id⟩/cancel [with JSON containing list of files]*
Deletion of a bulk-request	-	*DELETE /api/v1/stage/⟨request_id⟩*

As mentioned before, the EOS+CTA implementation of XRootD staging is file-based while the HTTP Tape REST API is bulk request based. However, it's important to note that XRootD does not necessarily forbid the usage of the *Request ID* for Bulk Requests. The decision not to implement *Request ID* to *File* mapping was a design choice from EOS+CTA.

In the EOS+CTA XRootD implementation, the *Request ID* is simply compared against the value previously stored in the attributes of each file that was referenced by the request. This allows EOS+CTA to validate that the file was indeed affected by that request.

Since bulk-request resources do not exist in the EOS+CTA XRootD implementation it has no concept for deleting them.

For more details see also the specification document, section *STAGE* [14].

2.3 Releasing staged files

Both protocols allow the client to indicate that they no longer require disk-like latency for the staged files.

Use case	XRootD	HTTP Tape REST API
Evict tape-stored disk copies from the disk buffer	*xrdfs prepare -e /path/file1.txt /path/file2.txt ...*	*POST /api/v1/release/⟨request_id⟩ [with JSON containing list of files]*

In the EOS+CTA implementation of XRootD the *request ID* is no longer associated with the file after its staging has been completed. Therefore, only the list of file paths needs to be passed.

In the HTTP Tape REST API the request persists until it has been explicitly deleted. Therefore, a release request to it only requires the *request ID*.

For more details see also the specification document, section *RELEASE* [14].

2.4 Endpoint mechanism

In addition to the previous specifications, the HTTP Tape REST API requires the servers to have a discovery mechanism, which allows the clients to find the URI endpoints of the API (reference document, section *Tape REST API discovery mechanism* [14]). It is defined that the *.well-known* endpoint is found by taking the WebDAV endpoint with the absolute path */.well-known/wlcg-tape-rest-api*.

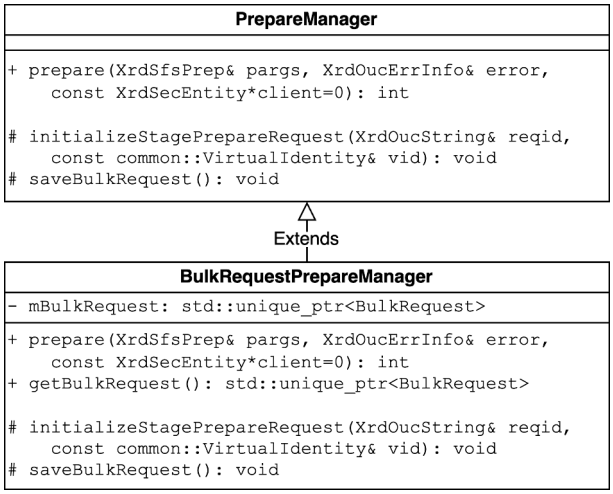

Figure 1: `BulkPrepareManager` class, which overrides some of the `PrepareManager` methods – while reusing most of its functionality—in order to implement the support for Bulk Requests.

3 Implementation in EOS+CTA

Considering that the handler for the XRootD *"prepare"* code already implemented most of the logic required to stage a file from tape to disk, it was our intention to reuse most of it. This was handled by the following `XrdMgmOfs` member function in the EOS mgm node:

```
int XrdMgmOfs::prepare(XrdSfsPrep& pargs,
        XrdOucErrInfo& out_error,
        const XrdSecEntity* client = 0) { ... }
```

This function was only meant for XRootD usage and could not be easily repurposed. In order to use it for the HTTP Tape REST API, we had to refactor it.

Our solution was to put the XRootD logic of the `prepare()` method into a class and split the code into different protected methods, each one for one different step of the staging process. Some of these methods could then be overriden by a subclass to implement the HTTP Tape REST API unique features, while most of the parent class code is simply reused.

In practice, this corresponds to the template method design pattern, which allows us to add functionalities to an already existing algorithm without having to change most of its behaviour. Only a small number of the methods would need to be updated to accommodate the new specifications. An example of this can be seen in Fig. 1.

Internally, depending on the protocol chosen by the client, either the implementation using `PrepareManager` or `BulkPrepareManager` will be invoked. Both of them will end up using the same Workflow Engine (WFE) code [19] to establish the event-based communication between EOS and the CTA Frontend, with the only difference that `BulkPrepareManager` is able to handle bulk requests. Fig. 2 shows how this is implemented.

The `BulkPrepareManager` implementation is invoked by the XrdHttp plugin of the XRootD framework.

3.1 Configuring the Tape REST API on EOS+CTA

In order to work, the HTTP Tape REST API needs to be configured on the EOS MGM node. This can be done by following the steps in the EOS online documentation [20].

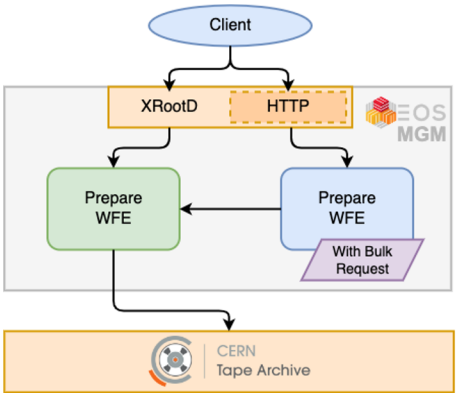

Figure 2: Diagram of Bulk Request implementation, which reuses most of pre-existing *prepare* functionality.

4 Deployment for LHC Run-3 operations

The HTTP tape REST API was requested in production by these two LHC experiments at WLCG Tier-0 (CERN):

- LHCb, which relies on HTTP archive transfers from WLCG Tier-1s to Tier-0. In particular, Rutherford Appleton Laboratory (RAL)—WLCG Tier-1 centre in the UK—needed the Tape REST API to be deployed for the LHCb HTTP staging campaign planned for February 2023.
- ATLAS, which is pushing to remove the dependency on SRM and a subsequent switch to WebDAV, in order to ease transfers between cloud storage and WLCG endpoints.

4.1 Tier-0 LHCb migration

The switch to the HTTP Tape REST API in the Tier-0 CTA production instances first took place on *eosctalhcb* (EOS+CTA instance for the LHCb experiment). Without the API LHCb Tier-1 to Tier-0 archive transfers were not being covered by the *Check-On-Tape* safety mechanism (checking that the file has a replica written to tape, therefore guaranteed to be safely stored in CTA).

A workaround was possible where DIRAC [21] would submit a multi-step archive job to EOS+CTA consisting of two jobs: (1) HTTP transfer to *eosctalhcb*; (2) XRootD transfer from *eosctalhcb* to itself with *Check-On-Tape* enabled, to confirmn that the file had been written to tape.

However, this was difficult to perform and was not systematically used in production. Switching to the HTTP Tape REST API made it possible to remove this complex multi-step workaround and to add *Check-On-Tape* to all Tier-1 to Tier-0 transfers, because finally a single protocol could be used for archival and *Check-On-Tape*.

In addition to the archive transfer logic simplification, Tier-1 transfers were sometimes performed using the XRootD protocol (XRootD 3rd Party Copy (TPC) transfers with X.509 proxy delegation). This kind of transfer required the deployment of XRootD TPC gateways for the *eosctalhcb* instance and introduced inefficiencies in the transfers. As a result, every transfer needed to go through a set of gateways.

Therefore, switching to the Tape REST API in production allowed us to offer a single efficient protocol that covers all CTA best practices.

The switch to HTTP with the TAPE REST API, in production, for the LHCb experiment, took place on 20 March 2023.

4.2 Tier-0 ATLAS migration

The ATLAS migration was a full protocol switch from XRootD to HTTP, which allowed it to preserve the previously supported functionalities. In fact, all ATLAS archive transfers were already covered by *Check-On-Tape*.

In this case the Tape REST API brought HTTP features similar to the XRootD protocol. CTA Tier-0 was the first ATLAS Storage Element to be fully migrated to a HTTP-only protocol.

After these two preparation steps, ATLAS moved all Tier-0 traffic to HTTP on 18 April 2023. We observed a full switch for both archive and retrieve transfers, with no loss of transfer efficiency. Overall, it was a smooth experience.

5 Conclusion

In this work we started by explaining the HTTP Tape REST API, a common HTTP REST interface for managing files stored on tape. This was the resulting work of the WLCG Tape REST API group, which aimed to define a new protocol to be supported by the main WLCG tape storage systems (EOS+CTA, dCache, StoRM) and data transfer clients (gfal2, FTS).

This framework was successfully concluded and is expected to be adopted by all these storage systems, in particular the one which was the focus of this paper: EOS+CTA.

For the case of EOS+CTA, we showed how the HTTP Tape REST API was implemented, by using the *XrdHttp* plugin of the XRootD framework, and reusing a large section of the previous XRootD *prepare()* code. The parallelism—as well as the differences—between the XRootD and HTTP implementations was also covered, with the observation that the most significant difference is the adoption of bulk requests by the HTTP Tape REST API implementation, whereas the XRootD implementation is only file-based.

Overall, the HTTP Tape REST API is a natural consequence of a recent shift to HTTP-centered protocols in the WLCG community. This contributes to closing the gap between WLCG and the more general online community. In a time where the WLCG data is expected to break new records in throughput and stored volume—and new tools are rapidly being developed by the physics and online communities—it becomes important to fully support HTTP for Tape operations. This will guarantee that EOS+CTA and other storage systems are fully aligned in their objective of being reliable and flexible data storage systems in the future of High Energy Physics.

5.1 Future work

In the future, it is expected that authentication and authorisation within WLCG will move from X.509 certificates (signed with VOMS) to capability-based JSON Web Tokens (JWTs). More information about these protocols can be found in [22]. Many transfers and tape stage operations are performed via FTS [12, 13]. Therefore, to remove the WLCG dependency on certificates, every component in the chain has to be adapted to token support, including both FTS and EOS+CTA.

The token landscape is for the most part finalised with regards to disk-to-disk transfers, with the involved players implementing the changes in preparation for the future WLCG Data Challenge 2024 [23]. Token discussions with regards to tape operations are still in the early stages. The process follows an iterative pattern and a number of upcoming revisions are foreseen. As it stands, tape operations with token authentication are envisioned much later, with discussion to pick up pace only after the WLCG Data Challenge 2024.

References

[1] *Worldwide LHC Computing Grid*, http://wlcg-public.web.cern.ch/, accessed: Sep. 2023
[2] *SRM reference webpage*, https://sdm.lbl.gov/srm-wg/, accessed: Sep. 2023
[3] A. Dorigo, P. Elmer, F. Furano, A. Hanushevsky, WSEAS Transactions on Computers **1**, 348 (2005)
[4] Forti, Alessandra, *Tpc activities - future plans*, https://indico.cern.ch/event/1089054/
[5] Davis, Michael, et al., *The CERN Tape Archive Beyond CERN*, in *CHEP* (2023), *to be published*, https://indico.jlab.org/event/459/contributions/11316/
[6] A.J. Peters, E.A. Sindrilaru, G. Adde, *EOS as the present and future solution for data storage at CERN*, in *Journal of Physics: Conference Series* (IOP Publishing, 2015), Vol. 664, p. 042042
[7] E. Cano, V. Bahyl, C. Caffy, G. Cancio, M. Davis, O. Keeble, V. Kotlyar, J. Leduc, S. Murray, *CERN Tape Archive: a distributed, reliable and scalable scheduling system*, in *EPJ Web of Conferences* (EDP Sciences, 2021), Vol. 251, p. 02037
[8] Litvintsev, Dmitry, et al., *The Bulk service and WLCG TAPE API support in dCache*, in *CHEP* (2023), *to be published*, https://indico.jlab.org/event/459/contributions/11350/
[9] P. Fuhrmann, V. Gülzow, *dCache, storage system for the future*, in *Euro-Par 2006 Parallel Processing: 12th International Euro-Par Conference, Dresden, Germany, August 28–September 1, 2006. Proceedings 12* (Springer, 2006), pp. 1106–1113
[10] Vianello, Enrico, et al., *A RESTful approach to tape management in StoRM*, in *CHEP* (2023), *to be published*, https://indico.jlab.org/event/459/contributions/11360/
[11] L. Magnoni, R. Zappi, A. Ghiselli, *StoRM: A flexible solution for Storage Resource Manager in grid*, in *2008 IEEE nuclear science symposium conference record* (IEEE, 2008), pp. 1971–1978
[12] Murray, Steven, et al., *FTS Service Evolution and LHC Run-3 Operations*, in *CHEP* (2023), *to be published*, https://indico.jlab.org/event/459/contributions/11313/
[13] E. Karavakis, A. Manzi, M.A. Rios, O. Keeble, C.G. Cabot, M. Simon, M. Patrascoiu, A. Angelogiannopoulos, *FTS improvements for LHC Run-3 and beyond*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 04016
[14] *Tape REST API Reference Document*, https://github.com/wlcg-storage/wlcg-tape-rest-api/blob/main/WLCG_TapeRESTAPI_Reference_Document_v1.pdf
[15] B. Bockelman, A. Ceccanti, F. Furano, P. Millar, D. Litvintsev, A. Forti, *Third-party transfers in WLCG using HTTP*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 04031
[16] A.P. Millar, O. Adeyemi, G. Behrmann, P. Fuhrmann, V. Garonne, D. Litvinsev, T. Mkrtchyan, A. Rossi, M. Sahakyan, J. Starek, *Storage for advanced scientific use-cases and beyond*, in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* (IEEE, 2018), pp. 651–657
[17] Patrascoiu, Mihai, *Srm - tape: gfal/fts code changes plan*, https://indico.cern.ch/event/967159/
[18] Millar, A. Paul, *Future uniform tape access*, https://indico.cern.ch/event/1006673/
[19] Davis, Michael, et al., *Eos+cta workflows: Tape archival and retrieval*, https://indico.cern.ch/event/985953/contributions/4238328/
[20] *Instructions: Enable Tape REST API on the MGM*, https://eos-docs.web.cern.ch/taperestapi/configuration.html
[21] F.e. Stagni, P. Charpentier, R. Graciani, A. Tsaregorodtsev, J. Closier, Z. Mathe, M. Ubeda, A. Zhelezov, E. Lanciotti, V. Romanovskiy et al., *LHCbDirac: distributed computing in LHCb*, in *Journal of Physics: Conference Series* (IOP Publishing, 2012), Vol. 396, p. 032104
[22] Bockelman, Brian, *Analysis Facilities AAI*, in *CHEP* (2023), *to be published*, https://indico.cern.ch/event/1230126/contributions/5315413/
[23] Mc Kee, Shawn, *Dc24 planning and near term activities*, https://indico.cern.ch/event/1301513/