A RESTful approach to tape management in StoRM

Enrico Vianello[1,*], *Federica* Agostini[1], *Laura* Cappelli[2], *Tommaso* Diotalevi[3], *Angelo* Galavotti[3], *Jacopo* Gasparetto[1], *Francesco* Giacomini[1], *Samuele* Lanzi[3], *Roberta* Miccoli[1], *Aksieniia* Shtimmerman[1], and *Marcelo* Vilaça Pinheiro Soares[1]

[1]INFN-CNAF, Bologna, Italy
[2]INFN Ferrara, Italy
[3]University of Bologna, Italy

Abstract. The STOrage Resource Manager (StoRM) service implements the SRM specification to recall files from tape. Although the SRM protocol has been successfully used for many years, its complexity has pushed the WLCG community to adopt a simpler approach, more coherent with current web technologies. The WLCG tape REST API offers a common HTTP interface which allows clients to manage disk residency of tape-stored files and observe the progress of file transfers to disk. In the context of the StoRM project developed at INFN-CNAF, the StoRM Tape service implements this HTTP interface. It is deployed as a standalone component and uses an NGINX reverse proxy as authentication engine and an Open Policy Agent (OPA) service as authorization policy decision point. This new service will be distributed at the INFN Tier-1 in order to coexist with the current StoRM deployments, in particular with the Grid-Enabled Mass Storage System (GEMSS).

1 Introduction

Recalling files from a tape storage has relied for many years on the Storage Resource Manager (SRM) protocol [1]. Although it was successfully used, its complexity and the gradual abandonment of GridFTP, in favor of data transfer and management via HTTP, pushed the WLCG [2] community to adopt a simpler approach, more coherent with current web technologies.

This contribution presents StoRM Tape, an implementation of the WLCG Tape REST API specification. Section 2 gives a brief explanation about the life-cycle of a file in a typical StoRM [3] deployment. Section 3 summarizes the main operations included in the new API specification. Section 4 and 5 detail the implementation and testing of StoRM Tape. Section 6 and 7 provide some future outlook and some conclusions.

2 File life-cycle in StoRM

Data access latency can be described as the amount of time necessary for data to become available. Within the SRM context, the access latency for a file is described with two different values: *Online*, the lowest possible latency, and *Nearline*, which can be improved to a lower

*e-mail: enrico.vianello@cnaf.infn.it

latency by staging the file on a disk buffer. Recalling a file from a tape storage means, in SRM terms, changing its latency from *Nearline* to *Online*. Only files in *Online* status can be read/transferred by clients.

The current StoRM Backend implementation is not able to directly recall a file from tape. This operation is managed by GEMSS [4]. GEMSS is a full Hierarchical Storage Management (HSM) system , integrating the IBM General Parallel File System (GPFS) [5], the IBM Tivoli Storage Manager (TSM) and the StoRM Backend, developed at INFN since 2008. Its primary advantages are the high level of reliability and the minimum management effort that is needed for daily maintenance. In the context of a tape-enabled storage area, the StoRM Backend is queried by GEMSS in order to retrieve all the file recall requests. A tape-enabled storage area works as a buffer for a GPFS file-set stored on tape. StoRM uses the extended attributes mechanism to store some metadata related to a file (e.g. the checksum value). This metadata can include also information about file locality and the progress of a recall from tape. Figure 1 summarizes the life-cycle of a file hosted on a tape-enabled storage area.

The life-cycle starts as soon as a user uploads a new file on the buffer. The file is stored on disk, so it has *Online* access latency. Newly created files are periodically migrated to tape, which means that a copy of that file is created on TSM. This is automatically done by GEMSS. When a predefined occupancy threshold is reached on the disk buffer, GEMSS starts to convert all the files not involved in an ongoing read data transfer to *stubs*. A stub is a file on the disk buffer that represents a file already migrated to tape, but has no content: its metadata are set in such a way that its size corresponds to the size of the migrated file and its block count is zero. Stubs represent files with *Nearline* access latency.

When a user requests to read a file which is *Nearline*, if there is no other recall in progress for that file, the request is forwarded to GEMSS via an internal endpoint and GEMSS takes care of recalling the file from tape. When the recall ends successfully, the data latency is set to *Online* and the file is ready to be read and/or transferred.

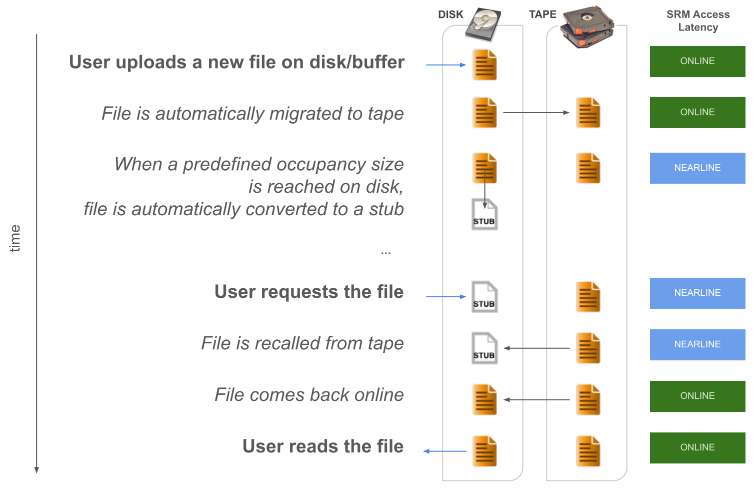

DISK
TAPE
SRM Access Latency
time
User uploads a new file on disk/buffer
ONLINE
File is automatically migrated to tape
ONLINE
When a predefined occupancy size is reached on disk, file is automatically converted to a stub
NEARLINE
STUB
...
User requests the file
NEARLINE
File is recalled from tape
NEARLINE
File comes back online
ONLINE
User reads the file
ONLINE

Figure 1. File life-cycle for a tape-enabled storage area in a typical StoRM deployment.

3 WLCG Tape REST API

The WLCG Tape REST API specification [6] has been defined within a working group including representatives from the storage providers, i.e. StoRM [3], dCache [7] and EOS+CTA [8]. The FTS project [9] was also involved, since it is the main client of the API.

The WLCG Tape REST API offers a common HTTP interface to manage disk residency of files stored on tape and to query their locality. The specification is designed to be storage-agnostic, so that the same protocol can be used to access a StoRM, dCache, or EOS+CTA tape storage system. The supported authentication and authorization mechanisms are based on X.509 certificates, VOMS (Virtual Organization Membership Service) [10] proxies and JSON Web Tokens (JWT) [11].

3.1 Specification

The Tape REST API specification supports bulk operations, which provide an optimization for clients and servers to work efficiently with a large number of files. The API supports the following operations:

- `POST /stage`: submits a bulk request of files to be recalled from tape to disk. The response includes a unique `id` which identifies the stage request
- `GET /stage/<id>`: tracks the progress of a previously submitted stage request
- `DELETE /stage/<id>`: deletes a previously submitted stage request
- `POST /stage/<id>/cancel`: declares that the recall of a subset of files belonging to a previously submitted stage request is not needed any more
- `POST /release/<id>`: declares that the disk residency of a subset of files belonging to a previously submitted stage request is not needed any more
- `POST /archiveinfo`: queries locality information about a set of files.

The SRM concept of *Online* or *Nearline* access latency is replaced by the concept of file locality in the Tape REST API specification. The locality values defined for a file are: *Disk* (the file is only on disk), *Tape* (the file is only on tape) and *DiskAndTape* (the file is both on disk and tape).

The specification also foresees a discovery mechanism which follows RFC 8615 [12]. The discovery endpoint must be implemented on each WebDAV site resolving the absolute path `/.well-known/wlcg-tape-rest-api`, which mainly indicates the Tape REST API endpoint. StoRM WebDAV supports the discovery endpoint pointing to the corresponding StoRM Tape instance since version 1.4.2 [13].

4 StoRM Tape

StoRM Tape [14] is the implementation of the WLCG Tape REST API for StoRM.

4.1 Implementation

StoRM Tape is a service written in C++, based on the Crow framework [15] and using the SOCI [16] library as abstraction layer over the SQLite database engine. The service checks the locality of files directly from the underlying storage system (GPFS). Information on files is obtained from the same extended attributes already used by other StoRM services and GEMSS.

StoRM Tape provides additional endpoints dedicated to GEMSS, to retrieve the queued stage requests. The responses of these endpoints have been aligned to those given by the StoRM Backend, to enable deployment scenarios including both the current SRM implementation and StoRM Tape.

4.2 Request flow

StoRM Tape is a standalone component (i.e. not included within StoRM WebDAV), packaged either as a Docker image [17] or as an RPM [18]. Authentication and authorization are handled by external open-source components: NGINX [19] and Open Policy Agent (OPA) [20], respectively. In addition to the common compelling reasons to use open-source software, e.g. a large community of developers and users who collaborate to inspect the software for security vulnerabilities and other issues, NGINX and OPA are highly adopted among the research communities. Figure 2 shows the interactions between the components mentioned above, together with GEMSS/TSM and the GPFS filesystem.

The flow of a stage request can be summarized as follows:

1. the user submits a stage request (`POST`) to the REST API by presenting a VOMS proxy or JWT token. The request is VOMS/TLS terminated by NGINX. NGINX successfully authenticates the user otherwise an Unauthorized HTTP response is returned

2. NGINX sends the authorization request to the OPA engine by including relevant fields extracted from the presented credentials and from the stage request

3. OPA makes the authorization decision and sends it back to NGINX. In case of negative authorization, NGINX returns a Forbidden HTTP response code. In case of successful authorization, the request is forwarded to the StoRM Tape backend service which adds the request to its internal database and returns the request identifier `id`

4. the response from the API is relayed to NGINX

5. the final response is returned to the user

6. GEMSS periodically retrieves the list of ready-to-recall files from the service.

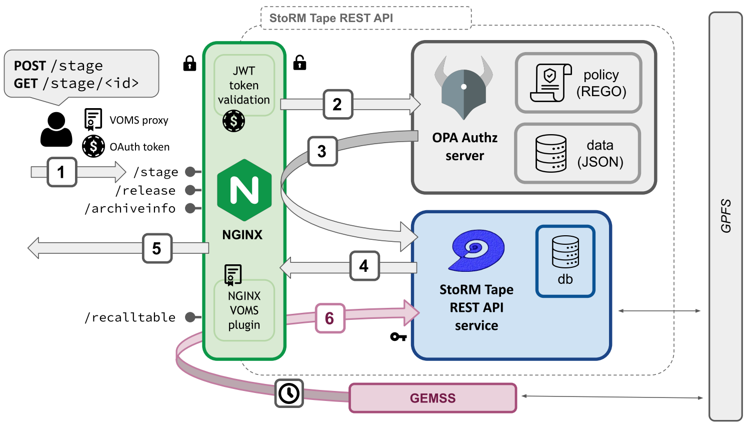


Figure 2. The StoRM Tape internal deployment schema. The steps of a stage request are highlighted from number 1 to 5. The interaction 6 is with GEMSS.

4.3 Internal components

4.3.1 NGINX

NGINX is an open-source HTTP server and reverse proxy, known for its high performance and stability, rich feature set and low resource consumption [21]. In the context of StoRM Tape deployment, it is used for TLS/VOMS termination and to support authentication with JWT tokens, X.509 certificates and VOMS proxies.

The authentication logic is driven by an NJS [22] script embedded in the NGINX configuration. The script first checks for the presence of a JWT bearer token in the HTTP Authorization header and validates it. Alternatively it checks for the presence of a VOMS proxy, through the NGINX VOMS module [23], in which case it makes the information (e.g. the VO name or the FQANs) available as NGINX variables. In case of a successful user authentication, an authorization request is sent to the OPA engine, including information extracted from the credentials. A successful response from OPA means that the original HTTP request can be forwarded to the StoRM Tape backend. If authentication or authorization fail, the user receives an Unauthorized or Forbidden HTTP response respectively.

In addition to the authentication flow described above, NGINX is also used to support the need of GEMSS to retrieve the list of ready-to-recall files from StoRM Tape. The access to this internal endpoint is restricted in multiple ways. GEMSS has also been properly modified in order to be able to contact both the StoRM Backend and StoRM Tape.

4.3.2 Open Policy Agent

Open Policy Agent is an open-source authorization engine that unifies policy enforcement across the stack, thanks to a high-level declarative language, called *rego*, that allows the definition of policies as code. OPA takes policy decisions by evaluating the query input against policies and data.

Since there is not yet a well-defined specification for the authorization rules of the WLCG Tape REST API, StoRM Tape implements the following coarse-grained authorization:

- `GET /stage/<id>` is accessible by a list of VOMS FQANs, X.509 subjects (useful when interacting through the browser) and JWT scopes with read permission
- all the other endpoints defined in Section 3.1 are accessible by a list of VOMS FQANs and JWT scopes with write permission
- otherwise HTTP Forbidden is returned.

A rego policy has been defined for all operations. The list of allowed VOMS FQANs, X.509 certificate DNs, scopes with read and write permissions is stored in a file in JSON format, owned by service operators. As an example, in the current proof-of-concept deployment, the allowed VOMS FQANs are `/wlcg/xfers` and `/wlcg/stage`, and the allowed X.509 certificate DNs belong to the people testing the service. Moreover, following the specification described by the WLCG Common JWT Profiles [24], we have assumed that read permissions are granted to `storage.read:/` and `storage.stage:/` scopes, and write permissions to the `storage.stage:/` scope. A more fine-grained authorization with JWT can be achieved when the parametric part of the scopes contains the path of the files to be recalled.

5 Testing

Beside the unit tests of the StoRM Tape code base, deployment tests [25] based on `docker-compose` and integration tests based on Robot Framework [26] have been performed. The Robot testsuite covers authentication and authorization, compliance with the functional specification and integration with GEMSS. At the time of writing, most tests complete successfully and the failures, concerning minor aspects, are being addressed.

To profile the performance of the StoRM Tape service itself, the code has been instrumented to collect tracing information. The output is a JSON file that can be imported and visualized in a browser (e.g. Chrome or Firefox). The observed average latency is less than 10 ms. This value is related to the interaction with StoRM Tape API and does not consider NGINX and OPA authentication and authorization layers, respectively. In order to measure the whole request duration, load tests using the Vegeta tool [27] have also been performed. Running 20 requests per second, the observed average latency of a stage request is below 100 ms. The tested request rate represents a reasonable request rate for the access of a tape based storage system. The latency value will be further improved by authentication and authorization layers optimizations (e.g. adopting JWT key caching) that are on the development road-map.

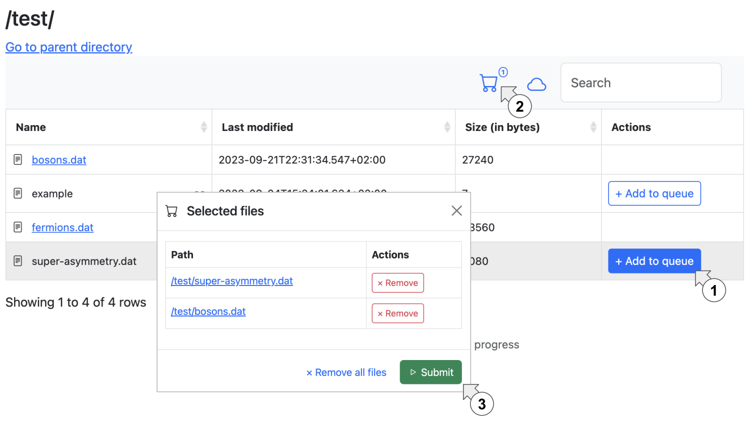
6 Future outlook

6.1 Developments and integration

The advantages of using the HTTP protocol for the Tape REST API is that it can be easily integrated into other services, for instance through a web interface.

StoRM WebDAV already allows users to navigate storage areas with a browser. An integration with StoRM Tape is under development and will allow some privileged users to trigger a stage request through the folder view, as shown in Figure 3. A list of privileged users can be identified by a JWT group membership or a DN, for example. It is also planned to provide additional endpoints to enable a monitoring dashboard for service operators.

The logs produced by StoRM Tape will eventually be sent to the Big Data Platform (BDP) deployed at INFN-CNAF [28]. The platform is based on Apache Kafka, used as a message broker, Logstash, which is a data processing instance, and OpenSearch, a data discovery and visualization engine.


/test/

Go to parent directory

Name	Last modified	Size (in bytes)	Actions
bosons.dat	2023-09-21T22:31:34.547+02:00	27240	
example			+ Add to queue
fermions.dat		3560	
super-asymmetry.dat		080	+ Add to queue

Showing 1 to 4 of 4 rows

Selected files

Path	Actions
/test/super-asymmetry.dat	× Remove
/test/bosons.dat	× Remove

Figure 3. Integration of StoRM WebDAV with StoRM Tape. A list of privileged users can submit a stage request through a web interface.

6.2 No-SRM scenario

The decommissioning of all Globus GridFTP transfer endpoints, replaced by instances of StoRM WebDAV, and the availability of an implementation of the WLCG Tape REST API are necessary steps towards a deployment scenario free of the SRM legacy (Figure 4).

The introduction of StoRM Tape provides a valid alternative to the current StoRM functionality and will allow tape-enabled storage areas in a no-SRM deployment. For StoRM, a no-SRM deployment means the removal of the StoRM Frontend and of a substantial part of the current StoRM Backend, keeping just the component dedicated to collect information about the storage areas (e.g. the used space).

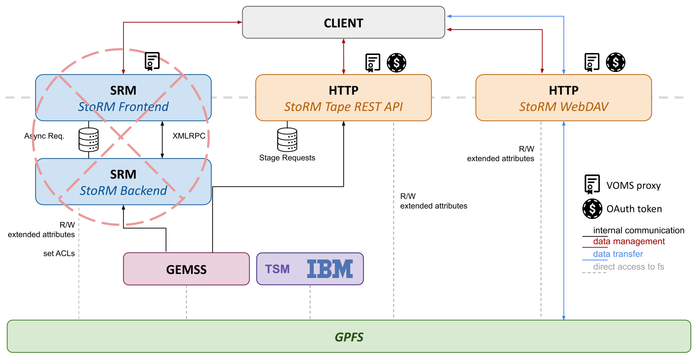


Figure 4. A no-SRM future StoRM deployment for a tape-enabled storage area.

7 Conclusions

The WLCG Tape REST API working group has defined a common HTTP interface which allows clients to manage access to files stored on tape and to observe the progress of file transfers on disk. The StoRM team has developed its own implementation of the specification, named StoRM Tape. Initially, this service will coexist with the current SRM deployment. StoRM Tape relies on NGINX and OPA to handle client authentication and authorization, respectively. VOMS proxies and JWTs are the currently supported authentication mechanisms. In a configured testbed agreed within the CNAF Storage group, NGINX and OPA have been properly setup and the first stage bulk requests have been executed successfully. Deployment and integration tests of the entire setup are carried out in a containerized environment. The StoRM Tape service appears to fulfill both performance and functional requirements and is planned to be moved to production at the INFN Tier-1 within the end of the year.

References

[1] *The Storage Resource Manager Interface Specification. Version 2.2*, `https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html`

[2] *Worldwide LHC Computing Grid (WLCG)*, `https://wlcg.web.cern.ch/`

[3] *STOrage Resource Manager (StoRM)*, https://italiangrid.github.io/storm, `https://doi.org/10.5281/zenodo.8088065`

[4] P.P. Ricci, D. Bonacorsi, A. Cavalli, L. Dell'Agnello, D. Gregori, A. Prosperini, L. Rinaldi, V. Sapunenko, V. Vagnoni, *The Grid Enabled Mass Storage System (GEMSS): the Storage and Data management system used at the INFN Tier1 at CNAF.* (2012), https://dx.doi.org/10.1088/1742-6596/396/4/042051

[5] F. Schmuck, R. Haskin, *GPFS: A Shared-disk File System for Large Computing Clusters*, in *Proceedings of the 1st USENIX Conference on File and Storage Technologies* (USENIX Association, Berkeley, CA, USA, 2002), FAST'02, pp. 16–16, http://dl.acm.org/citation.cfm?id=1973333.1973349

[6] *WLCG Tape REST API (v1) reference document*, https://docs.google.com/document/d/1Zx_H5dRkQRfju3xIYZ2WgjKoOvmLtsafP2pKGpHqcfY

[7] *dCache: Inter-disciplinary storage system*, https://doi.org/10.5281/zenodo.8129517

[8] *CERN Tape Archive (CTA)*, https://cta.web.cern.ch/cta

[9] *File Transfer Service (FTS)*, https://fts.web.cern.ch/fts

[10] *Virtual Organisation Membership Service (VOMS)*, https://italiangrid.github.io/voms, https://doi.org/10.5281/zenodo.1875371

[11] M.B. Jones, J. Bradley, N. Sakimura, RFC 7519, IETF Tools (2015), https://tools.ietf.org/rfc/rfc7519.txt

[12] M. Nottingham, *Well-Known Uniform Resource Identifiers (URIs)*, RFC 8615 (2019), https://www.rfc-editor.org/info/rfc8615

[13] *StoRM WebDAV latest release: v1.4.2* (2023), https://github.com/italiangrid/storm-webdav/releases/tag/v1.4.2, https://doi.org/10.5281/zenodo.8088068

[14] *StoRM Tape REST API source code*, https://baltig.infn.it/cnafsd/storm-tape

[15] *Crow source code*, https://github.com/CrowCpp/Crow

[16] *Simple Oracle Call Interface (SOCI) source code*, https://github.com/SOCI/soci

[17] *Container Registry for StoRM Tape REST API*, https://baltig.infn.it/cnafsd/storm-tape/container_registry/374

[18] *Package Registry for StoRM Tape REST API*, https://baltig.infn.it/cnafsd/storm-tape/-/releases

[19] *NGINX*, https://nginx.org/en/docs

[20] *Open Policy Agent (OPA)*, https://www.openpolicyagent.org/docs/latest

[21] D. Kunda, S. Chihana, S. Muwanei, *Web Server Performance of Apache and Nginx: A Systematic Literature Review* (2017)

[22] *NJS Scripting Language*, https://nginx.org/en/docs/njs/

[23] *VOMS module for NGINX*, https://baltig.infn.it/cnafsd/ngx_http_voms_module

[24] M. Altunay, B. Bockelman, A. Ceccanti, L. Cornwall, M. Crawford, D. Crooks, T. Dack, D. Dykstra, D. Groep, I. Igoumenos et al., *WLCG Common JWT Profiles* (2019), https://doi.org/10.5281/zenodo.3460258

[25] *Tape rest API testsuite*, https://baltig.infn.it/cnafsd/storm-tape-ts

[26] *Robot Framework User Guide*, https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html

[27] *Vegeta*, https://github.com/tsenart/vegeta

[28] S. Rossi Tisbeni et al., *A Big Data Platform for heterogeneous data collection and analysis in large-scale data centres* (2021)