XrdClHttp

XrdClient

XrdCl

TFile::Open()

Back on the case

F.Furano

XROOTD workshop 2026

DAVIX

TNetXNGFile

TDavixFile

XrdClS3

XTNetFile

The topic

- High performance, scalable, flexible data access for the ROOT framework
- ROOT clients and various kinds of servers
- The protocol wars
  - HTTP
  - WebDAV
  - Xrootd
  - S3
  - MS Azure

Genealogy

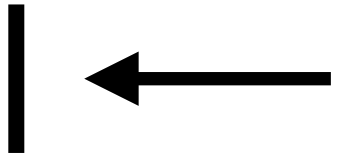
- 2004: CHEP conference: “XTNetFile, a fault tolerant extension of ROOT TNetFile” (Dorigo, Furano, Elmer, Hanushevsky)
- Later 2004: Code separation, TXNetFile (ROOT) becomes a wrapper. Rewriting of XrdClient and xrdcp
- 2013: ROOT workshop
  - "DAVIX: a client library for WebDAV and HTTP-derived protocols oriented to high performance data and metadata access" (Furano, Devresse)
  - [TNetXNGFile] "Features of the new XRootD client and ROOT IO plugin" (Łukasz Janyst)
- GDB - Jan 2014: “Bring Davix HTTP support to ROOT I/O (TDavixFile)” (Devresse)
- ... 12 years of relative stability

The protocols

- Xrootd: sophisticated for high performance data access
  - X509, X509proxy, UNIX, Kerberos, Various tokens
- HTTP
  - The Web standard for downloading pages and files. Supports random access and multirange
    - X509, X509proxy (only in some servers), Various tokens
- WebDAV
  - HTTP-based, better support for metadata and file listings
- S3
  - The industry standard for Cloud storage. HTTP based. Can do random access (depending on the server). No multirange.
    - Proprietary V2 and V4 signatures, documented
- MS Azure
  - Very similar to S3, crafted by Microsoft. HTTP based.
    - Proprietary signatures, documented

On the ROOT TFile:: side

- TNetXNGFile uses XrdCl (from the Xrootd codebase, in GitHub). XrdCl can load “protocol plugins”
  - Xrootd
  - HTTP
  - S3



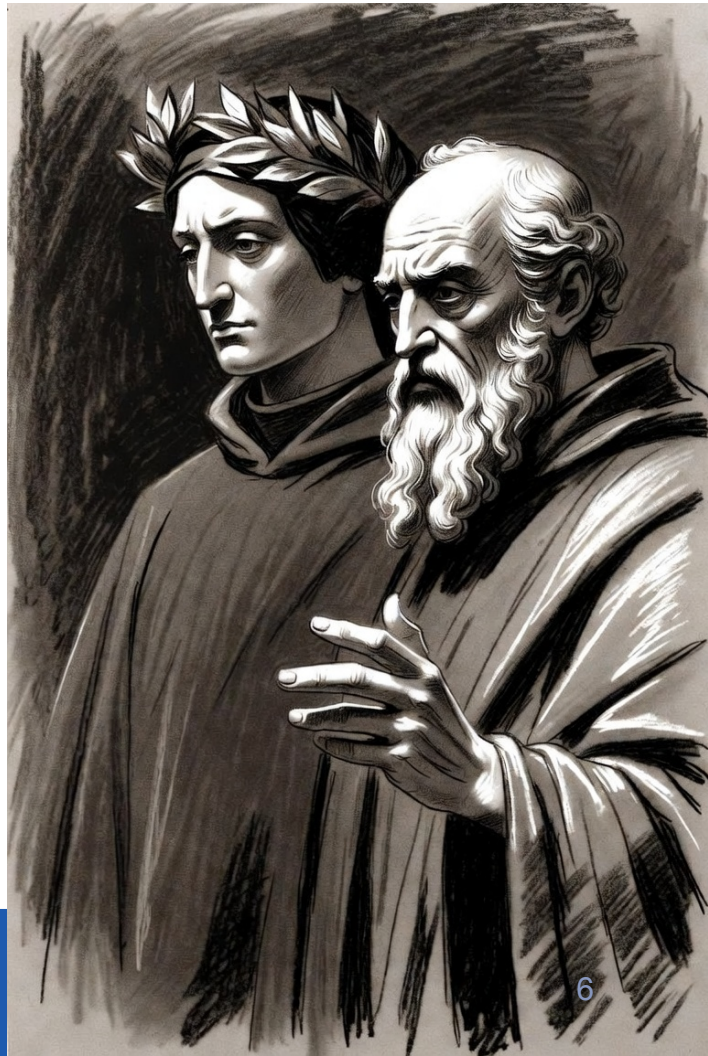
NEW contribution (2026)

- TDavixFile uses Davix (standalone library, in GitHub)
  - HTTP
  - WebDAV
  - S3
  - Azure

- Regular ROOT users do not see this, thanks to the TFile plugin manager
  - **`TFile::Open( url, ... )`**

Direction

- Using only XrdCl [TNetXNGFile] for ROOT data access
  - Xrootd
  - HTTP
  - S3 in future releases
- Decrease support for Davix
- … while other working groups work on using XrdCl (instead than Davix) on what will replace GFAL2



TFile::Open()

- ROOT loads *at runtime* the right library to access a file

```
TFile *f = TFile::Open("root://myxrd.cern.ch:1094//h1.root");
TFile *f = TFile::Open("https://myxrd.cern.ch:1094//h1.root");
TFile *f = TFile::Open("s3://mys3.cern.ch//h1.root");
```

- Where root TEnv parameters define how to authenticate, certificates, etc.
- The TDavix* parameters and XNet* parameters are not the same
- Code that used to work with one will need to be modified to work with the other
- At least we need to minimize the burden making things look clean and analogous

ROOT integration - steps

- Make sure TNetXNGFile can handle root:// and http[s]:// and s3[s]:// URLs
- Make the rootrc configurations look similar for TDavixFile and TNetXNGFile (mainly security configurations to be made compatible, i.e. cover the same use cases)
- Make sure that the common auth cases work (e.g. X509, Proxy certs, …)
- Heavy debugging, create a stable starting point for future usage and evolution

ROOT integration - layers

- These steps concern
  - ROOT wrapper classes
  - Core functionalities of XrdClHttp and XrdClS3

- The ROOT wrappers are complete [Sept 2026]
  - Expect WIP PR these days

- The core functionalities of XrdCl[Http|S3] are there, the basics work
  - Some bad bugs of the multipart reads have been fixed
    - Great interplay with Rahul Chauhan
  - Also the historical TTreeCache tests work

- These assume a path of improvement, development and maintenance on the XrdCl* side
    - Implement the bits that are missing
    - Keep everything shiny and working

A look at these params

- The overall structure is traditionally, verbosely documented in the rootrc file

```
XSec.GSI.GridMode
XSec.GSI.CADir
Xsec.GSI.CAFile
XSec.GSI.VerifyPeer
XSec.GSI.UserCert
XSec.GSI.UserKey
XSec.GSI.UserProxy

NetXNG.S3.XrdClS3MkdirSentinel
NetXNG.S3.XrdClS3Endpoint: the hostname of the S3 service, e.g. "s3.cern.ch"
NetXNG.S3.XrdClS3UrlStyle: "path"->the bucket name is in the path "virtual"->the bucket name is in the
hostname
NetXNG.S3.XrdClS3Region:
NetXNG.S3.XrdClS3BucketConfigs: A list of lines containing bucket names. Credentials for these buckets
will be fetched from properly named config files. See the XrdClS3 documentation.
NetXNG.S3.XrdClS3AccessKeyLocation: absolute path to a file that contains the S3 access key
NetXNG.S3.XrdClS3SecretKeyLocation: absolute path to a file that contains the S3 secret key
NetXNG.S3.SecretKey: secret
NetXNG.S3.AccessKey: access
NetXNG.S3.Token: token or empty
```

Notable open points (1/2)

- S3 from ROOT (readonly)
  - Davix supports it, XrdClS3 does not
    - Random access will need to be implemented in XrdClS3
    - S3 optimizations, e.g. vector reads range coalescing makes a big performance difference in Davix
    - XrdClS3 has its own clever mechanisms for managing sets of S3 keys in the filesystem, it needs also to support the “simplest” direct passing of the keys, like Davix

- WebDAV
  - Davix can interact using WebDAV, for complete metadata management, or with the simplest possible HTTP. Depending on the URL “http://“ or “dav:”
  - XrdClHttp assumes WebDAV, or parts of it just with the URL pfx “http://“
    - Not all the functionalities will work if the server is too basic (e.g. a vanilla Apache)
    - File stat and listings may need to be thoroughly verified for the needed cases

Notable open points (2/2)

- DAVIX Redirection caching [impact to be understood]
- DAVIX Parallel multirange requests [impact to be understood]
- Various flavours of https, e.g. anonymous, X509, X509 proxy [almost there]

- Complex usage patterns for bearer tokens. Are we sure we are considering all those use cases? [I am not]

- Let's not forget TDavixSystem/TNetXNGSystem
  - Ongoing, much simpler than TNetXNGFile

Conclusion - Or beginning

- Delicate software shuffling
- Declared outcome is to allow using XrdCl instead than Davix
- NB Davix is already in EPEL10, up to 2030s
- New ROOT integration layer is coming as we speak
  - Setting it as default will be a separate action on the ROOT side

- Some XrdClS3/XrdClHTTP improvements still needed
- We are giving a clear, stable, working starting point
  - What has to follow is improving and maintaining it