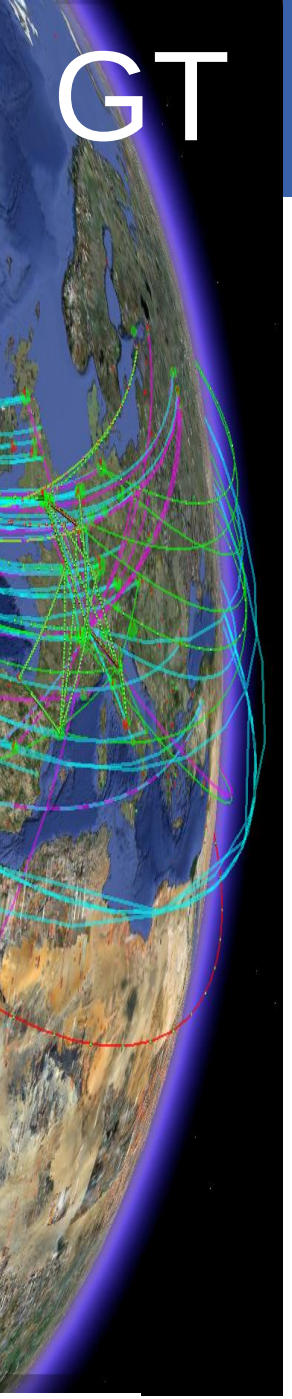
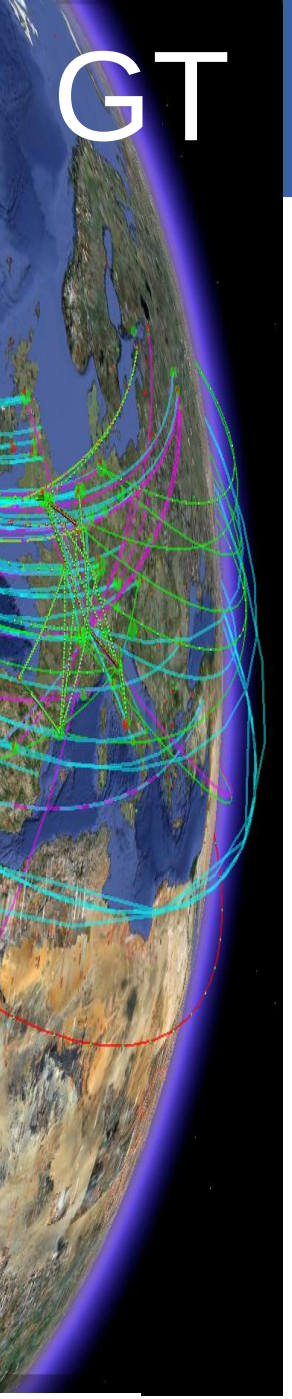


Davix

A toolkit
for efficient data access
with HTTP/DAV based protocols

Fabrizio Furano
Adrien Devresse

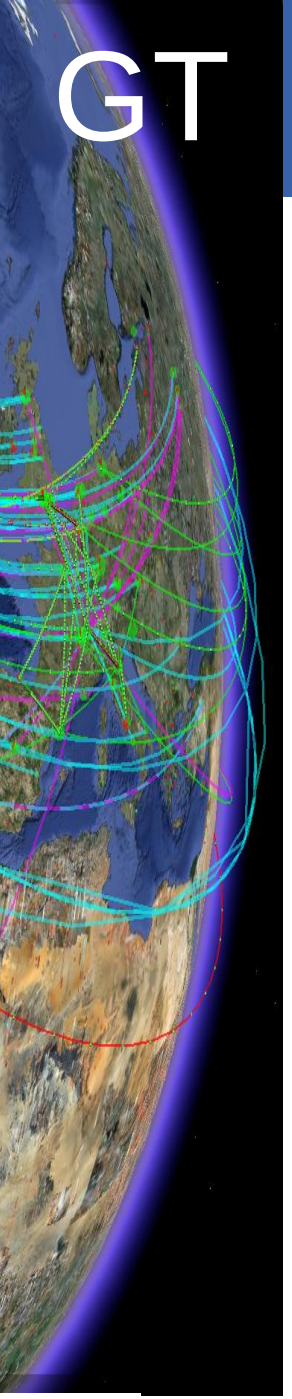
CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



The starting point

- Considerable activity in EMI around HTTP/DAV
- Outcome: the Grid Storage Elements support it
- Consensus about HTTP/DAV being a desirable goal for HEP data access
  - Our contributions go in this direction
- Services in general are seen through clients, over which applications are built
- The perception of such a standards-based service hence depends also on the quality and the features of the client that an user decides to use
  - To just d/l a file or
  - use a sophisticated I/O framework (ever heard about ROOT?)
  - use an existing application (e.g. a browser) or
  - design a new application

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



From services to applications

- These new-generation Grid standards-based systems are very feature-rich
  - Moreover, they are backward-compatible

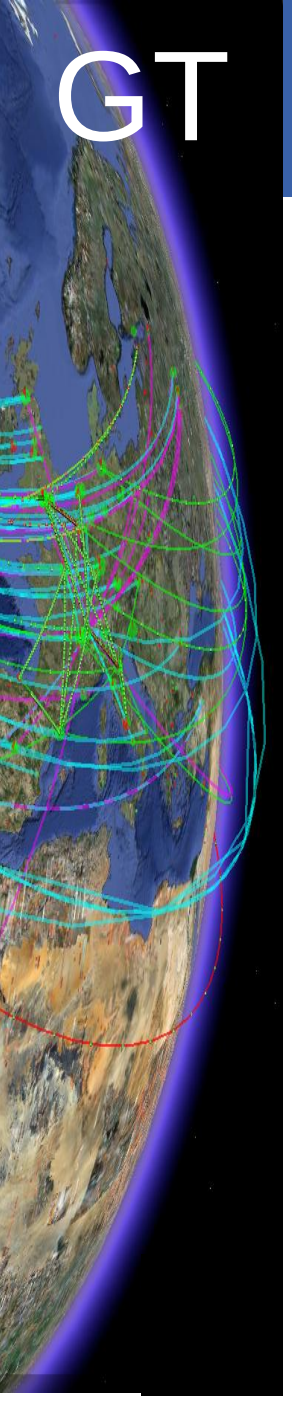
- We have been the first users of our own systems
- Standard clients always work with the new services.
- The user of the client sees basically the features of the client
  - These may just be as small as “returning an text snippet” that must be interpreted correctly for the transaction to be successful, e.g.
    - a redirection from https to http with a security token
    - a redirection while performing a PROPFIND or PUT (not supported by mainstream clients)
    - a credentials delegation
    - a response to a vectored read request

- What about dialects then? Systems that have to be queried in different ways to do the same thing, and will also answer differently e.g. S3 or DFS

- The risk is to give to the application developers the responsibility of interpreting things that are subtly storage-dependent, in the logic or in the implementation

- Our systems DO give advanced functionalities, we will be happy if everybody uses them!
- **These high level operations will be used at their best if they are correctly encapsulated and given as usable features of a complete client.**

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



A data mgmt client for DAV

- Users just want to access and manage their data. We are already providing server-side systems that are based on DAV.
- Users should not loose time in constructing HTTP/WebDAV/S3 queries and parsing complex responses.
  - The complexity of the code grows very fast as the features grow
  - This also multiplies the space for problems...bugs, inefficiencies, etc.
- The use case of the Storage Elements is I/O and metadata
  - We just wanted a fast, simple and reliable I/O API, with all the features.
  - We were not able to find a full-featured client already made for us.
- As users, we preferred to avoid doing and redoing implementations of sophisticated features.
- Our first use case was the Dynamic HTTP federations project, we started there the implementation of a full-featured client only once and for all.

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

A very basic example (1/2)

Simplified diagram for a stat() call using DAV, no security, no anything else

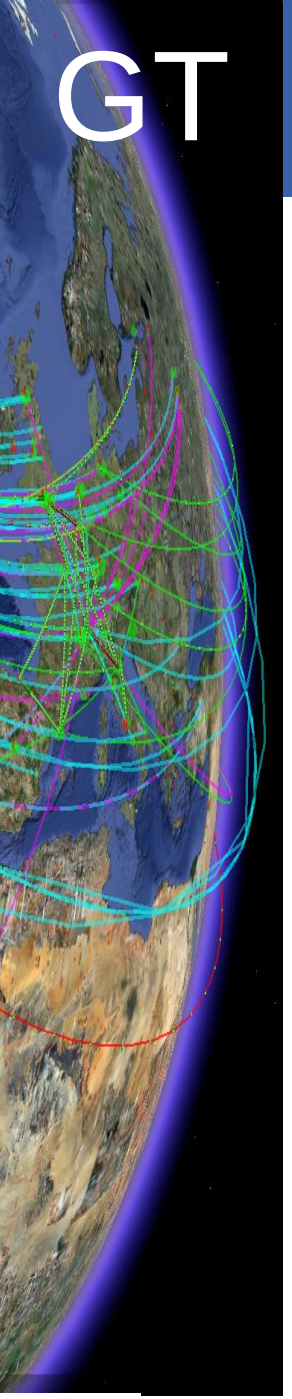
Request

```
PROPFIND /myfed HTTP/1.1
User-Agent: libdavix/0.0.9 neon/0.0.29
Connection: TE, Keep-Alive
TE: trailers
Host: federation.desy.de
Depth: 0
```

Response is any possible dialect of

HTTP/1.1 207 Multi-Status
Date: Fri, 01 Mar 2013 13:14:03 GMT
Server: Apache/2.2.15 (Scientific Linux)
Vary: Accept-Encoding
Content-Length: 954
Connection: close
Content-Type: text/xml; charset="utf-8"

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
<D:response xmlns:lcgdm="LCGDM:" xmlns:lp3="LCGDM:" xmlns:lp1="DAV:"
xmlns:lp2="http://apache.org/dav/props/">
<D:href>/*myfed*/</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype><D:collection/></lp1:resourcetype>
<lp1:creationdate>1970-01-01T00:00:00Z</lp1:creationdate><lp1:getlastmodified>Thu, 25 Oct
2012 14:33:00 GMT</lp1:getlastmodified><lp3:lastaccessed>Thu, 01 Jan 1970 00:00:00
GMT</lp3:lastaccessed><lp1:getetag>0-
50894d9c</lp1:getetag><lp1:getcontentlength>0</lp1:getcontentlength><lp1:displayname>/</lp1:
displayname><lp1:iscollection>1</lp1:iscollection><lp3:guid></lp3:guid><lp3:mode>040777</lp
3:mode><lp3:sumtype></lp3:sumtype><lp3:sumvalue></lp3:sumvalue><lp3:fileid>0</lp3:fileid>
<lp3:status>-
</lp3:status><lp3:xattr>{}</lp3:xattr><lp1:owner>0</lp1:owner><lp1:group>0</lp1:group></D:pr
op>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
</D:multistatus>



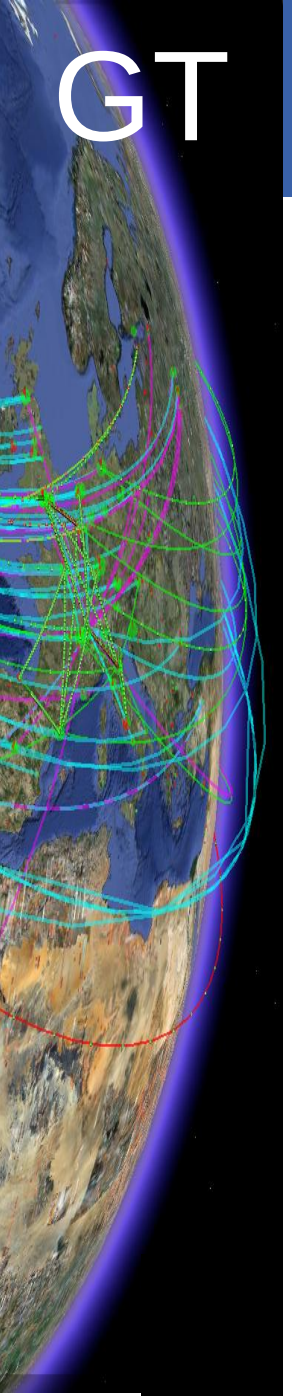
A very basic example (2/2)

- Our preference is to write instead:
  - `st = mydavclient.Stat("filename")`
  - ...and let XML work for us, not against.

- Writers of apps do not need to become DAV gurus and reimplement it
- The most sophisticated feature or dialects handling is properly encapsulated
- All the work behind being standards-compliant is done once and for all
- Encapsulating means making space for doing things at a sufficient level of performance and quality

- Again, this was just a simple example. Real things are more complex.

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



The basic bricks

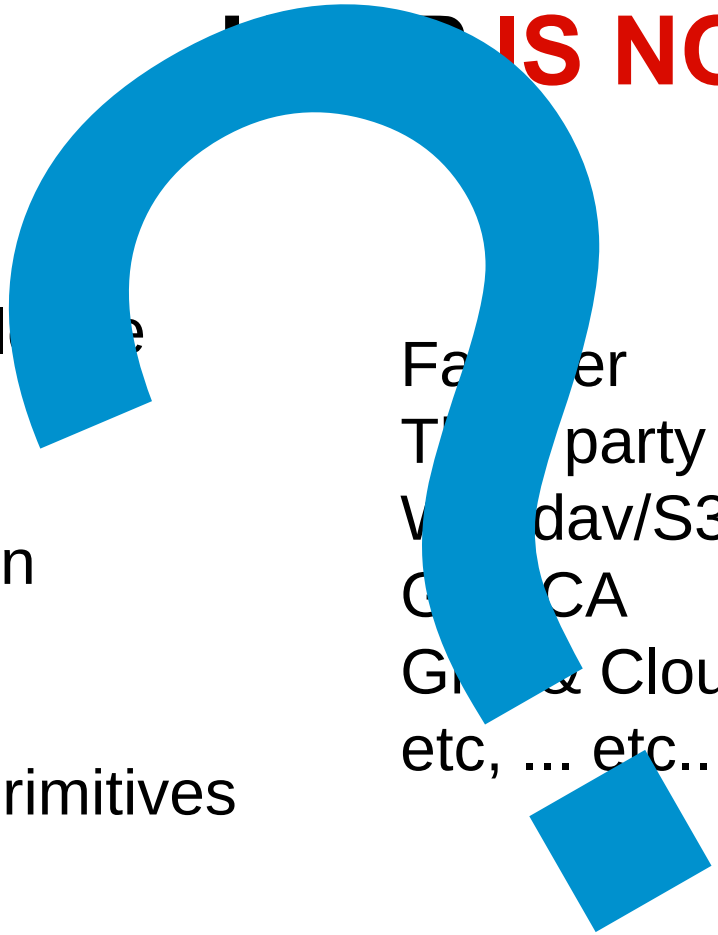
- For clusterization and HEP access to work we need some basic features:
  - Redirection support in all the primitives of HTTP and DAV
  - Flexible support for the various security protocols (e.g. GSI proxies, krb, pwd, x509)
  - Map the “posix-like” things (open, stat, close, ...) into HTTP/DAV requests/responses
- **Data access, data xfer and federation CAN be done with standard clients**
  - **We all agree.** At the same time it’s difficult to find elsewhere a complete HEP production level storage/data client that does what seems obvious to us.
  - The features that the users see are also proportional to the completeness of the clients they use and to their quality.
  - Forcing app developers to implement features over XML parsing is questionable. This is the option without a complete client.
  - We can use a standard client to implement what we need, on top of it. We used libneon at the bottom of our architecture.

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

I/O with HTTP...

Doing I/O with HTTP seems trivial...

Making efficient I/O with IS NOT trivial !



Random I/O vs whole
Vectored read/write
X509 authentication
VOMS authentication
Session re-use
Multi-stream xfers
Redirection in ALL primitives
Metalinks

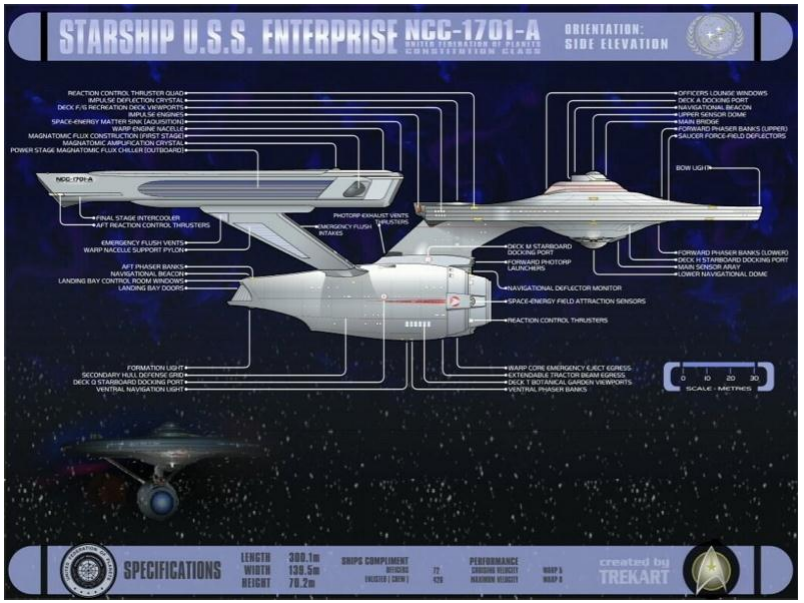
Fa er
T party copy
V dav/S3/DFS dialects parsing
C CA
G & Cloud extensions
etc, ... etc...

Design goals of Davix

Polyvalent

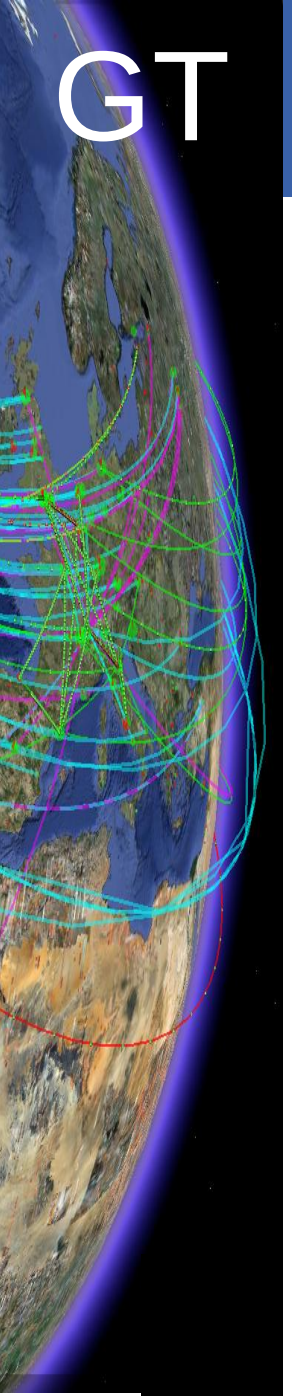
Lightweight

Easy to Use



Performant

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



Davix: A multi-layer API

- Object Store API
  - GET / PUT, Cloud like
- Posix-like API
  - open / read / close
  - opendir / readdir/ …
  - easy to use
- Low level API
  - Construct your query if you wish

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Features: security

- A complete authentication framework
- Client cert X509
  - PEM
  - PKCS12
  - VOMS proxy
  - VOMS extensions
  - Globus proxies
- Grid CA :
  - define your own CA path.
- Login/password
- S3 auth (*)
- Kerberos
- **SSL Session re-use**

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Features: Posix-like I/O

- High level meta-data functions
  - stat, mkdir, rmdir, unlink, opendir
  - delete, move, etc, etc...
- Full-file I/O - GET/PUT
- Partial I/O
- Vectored I/O
- Redirection on ALL operations
- Third party copy

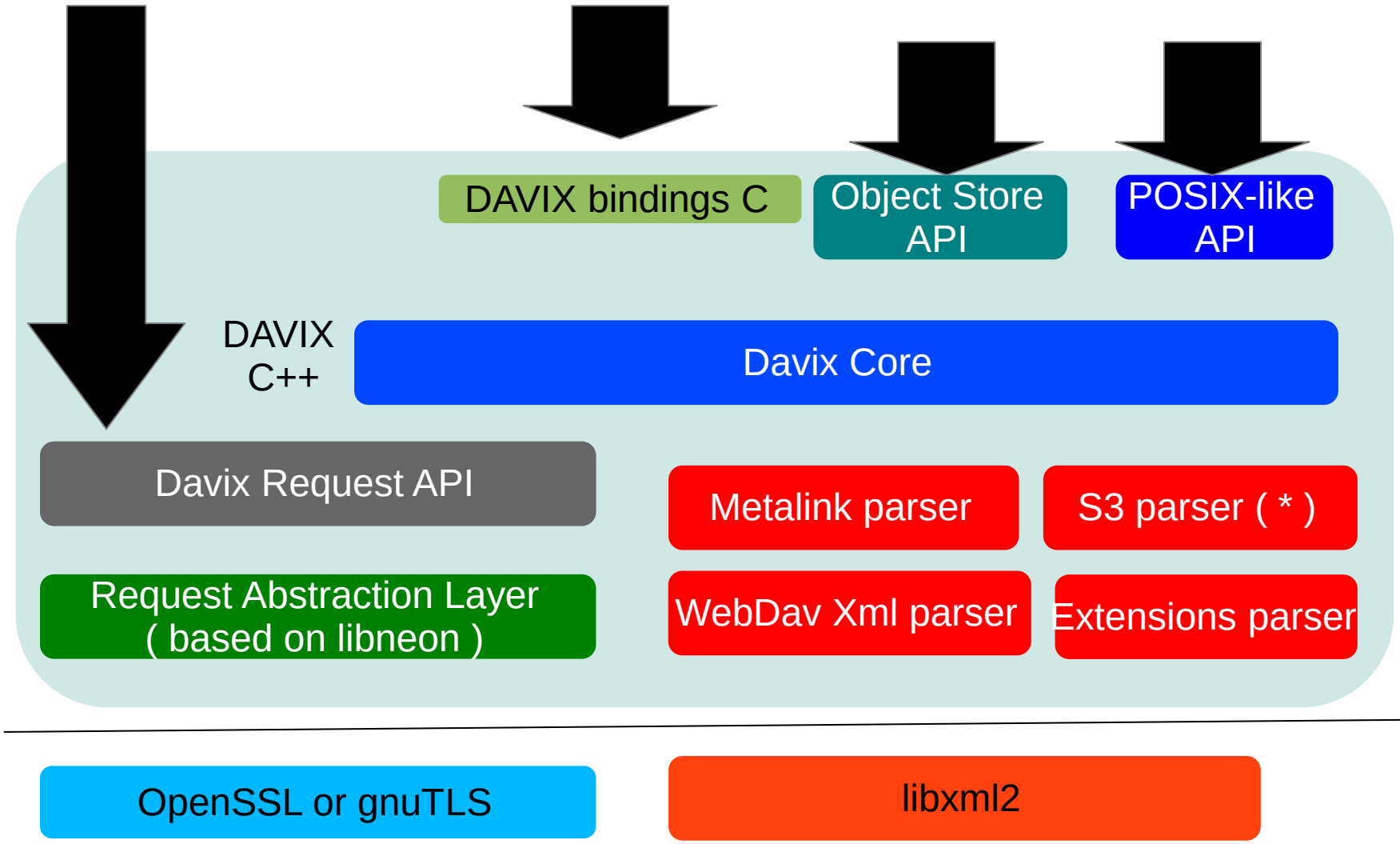
CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

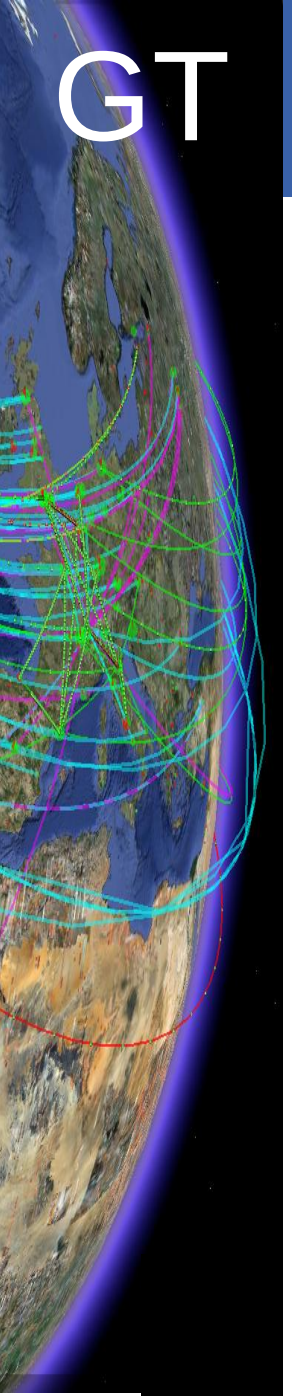
Features: Replica mgmt

- Replica management and failover :
  - Metalink and LCGDM DAV extensions
  - Transparent fetch of replica info to drive actions
  - Replica info drives transparent failover actions
    - Set the number of retries
- Multi-stream xfers fed by the same replica info
- Integrated with HTTP Federations.

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Davix Architecture

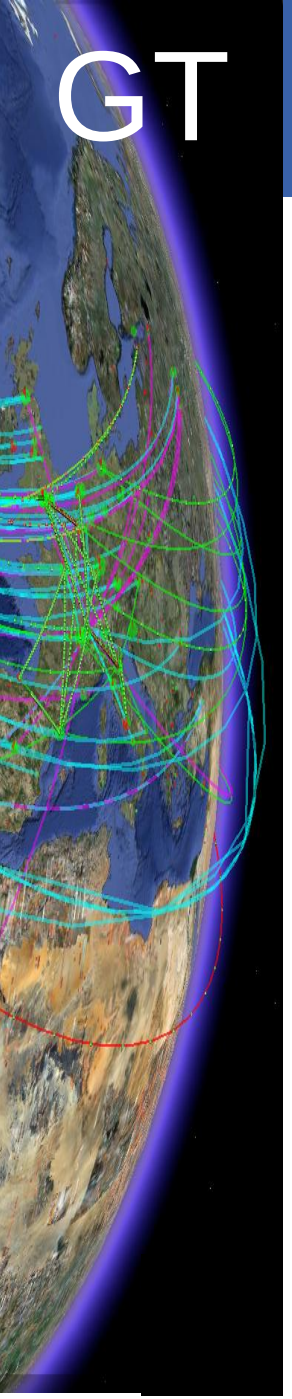




Tested

- Tested towards:
  - LCGDM-DAV (The DAV interface to the legacy DPM/LFC components)
  - STORM Webdav
  - DMLITE-DAV (The DAV interface to the new DPM/LFC based on DMLITE)
  - HTTP Dynamic Federations
  - DCACHE DAV door
  - Apache Webdav
  - CERN DFS new
  - Standard Python DavServer
  - Deutsche Telekom Cloud services
  - Amazon S3 is coming new
- Already used in :
  - HTTP Dynamic Federations
  - GFAL 2.0 as HTTP/DAV plugin
  - FTS 3.0 file transfer

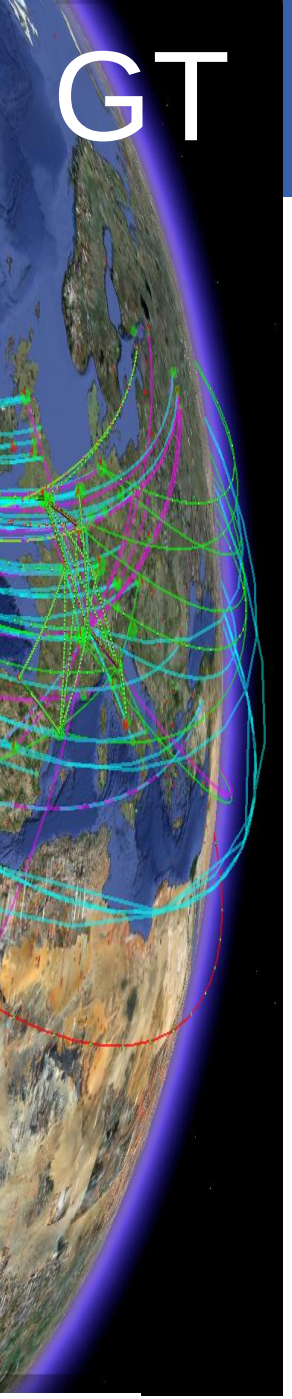
CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



Summary : What is Davix ?

- A lightweight, portable and reliable client data mgmt and I/O Library for HTTP-based protocols.
- Focused on performance for data access and data management in grids / clouds.
- Supports everything that you want to use and don't want to support
- Multi-layer API : Simplify life of user, give them power if they want it.
- A set of tools for Http/Webdav data management.

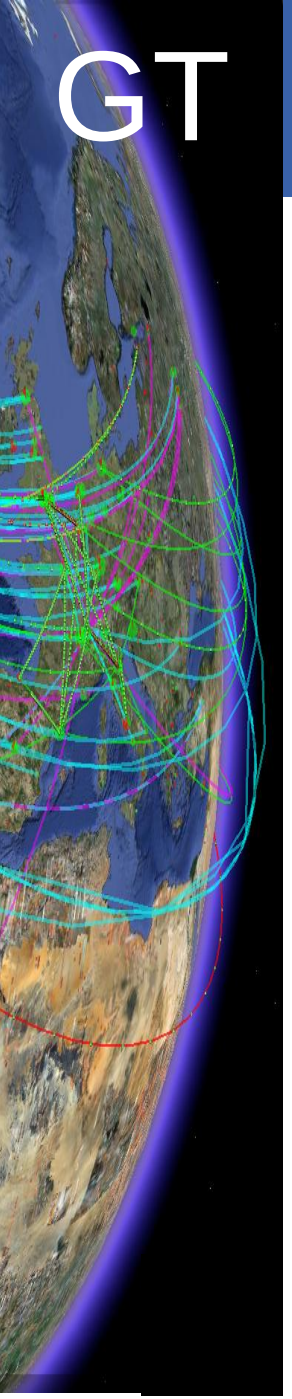
CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



ROOT: TWebFile/TWebSystem

- ROOT I/O can be extended by providing subclasses of TFile and TSystem.
  - http://root.cern.ch/root/html/TFile.html
  - http://root.cern.ch/root/html/TSystem.html
- The current one for HTTP is TWebFile (and TWebSystem has empty implementation)
  - http://root.cern.ch/root/html/TWebFile.html
  - http://root.cern.ch/root/html/TWebSystem.html
- An example of a full implementation is TXNetFile and TXNetSystem
  - http://root.cern.ch/root/html/TXNetFile.html
  - http://root.cern.ch/root/html/TXNetSystem.html

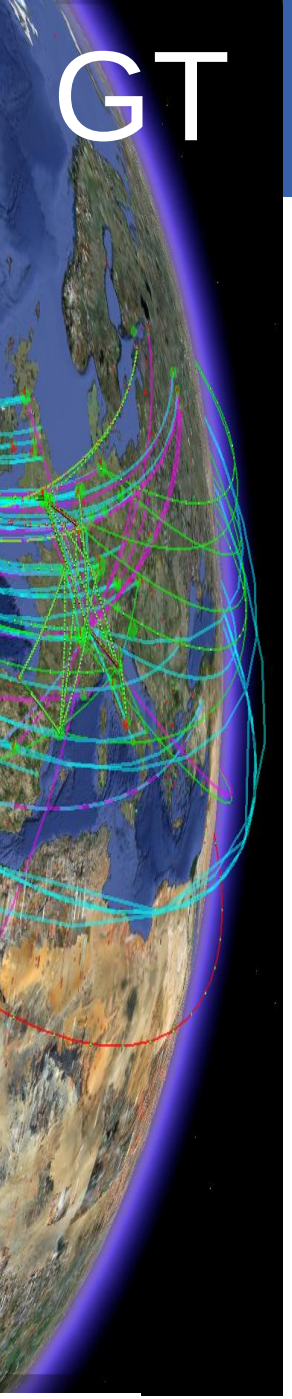
CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



ROOT: TWebFile/TWebSystem

- TWebFile: direct data access features for HTTP
  - Leaves the need for:
    - Grid auth[n/z] protocols: GSI, X509, proxy delegation, pwd
    - Auth session reuse (**makes a big difference in performance**)
    - Failover and request retry
    - Metalinks can give advanced features to failover
    - HTTP 503 response: wait and retry
- TWebSystem: the metadata features
  - In the case of a DAV server there are many features that can be given, e.g. file listings or operations on files like rm, mv, etc.
- We are agnostic about the existence of classes in ROOT that are specialized for a particular dialect (e.g. S3 or DFS). Anyway DAVIX is supposed to work when talking to those systems.
- As users, we think that a proper, unique implementation could offer a higher degree of interoperability.
  - It's very difficult to know in advance if an HTTP URL will speak a dialect or another. The implementation must be able to do it, and correctly map the features. This is what we do (among other things).
  - There is no reason to split clients for systems that are so similar, like DAV and S3.

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it



The current status

- Chats and meetings among IT-GT, PH-SFT and IT-DSS
- Likely ROOT will be asked for the advanced HTTP/DAV features
- We can provide our client to any group that wants to use it to provide HTTP/DAV client features to ROOT. Of course we do support it.
- Some people are exploring this (Fabio, Tigran)
- Would be a component shared by GFAL2, FTS3, UGR and ROOT

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it

Questions ?

CERN IT Department
CH-1211 Geneva 23
Switzerland
www.cern.ch/it