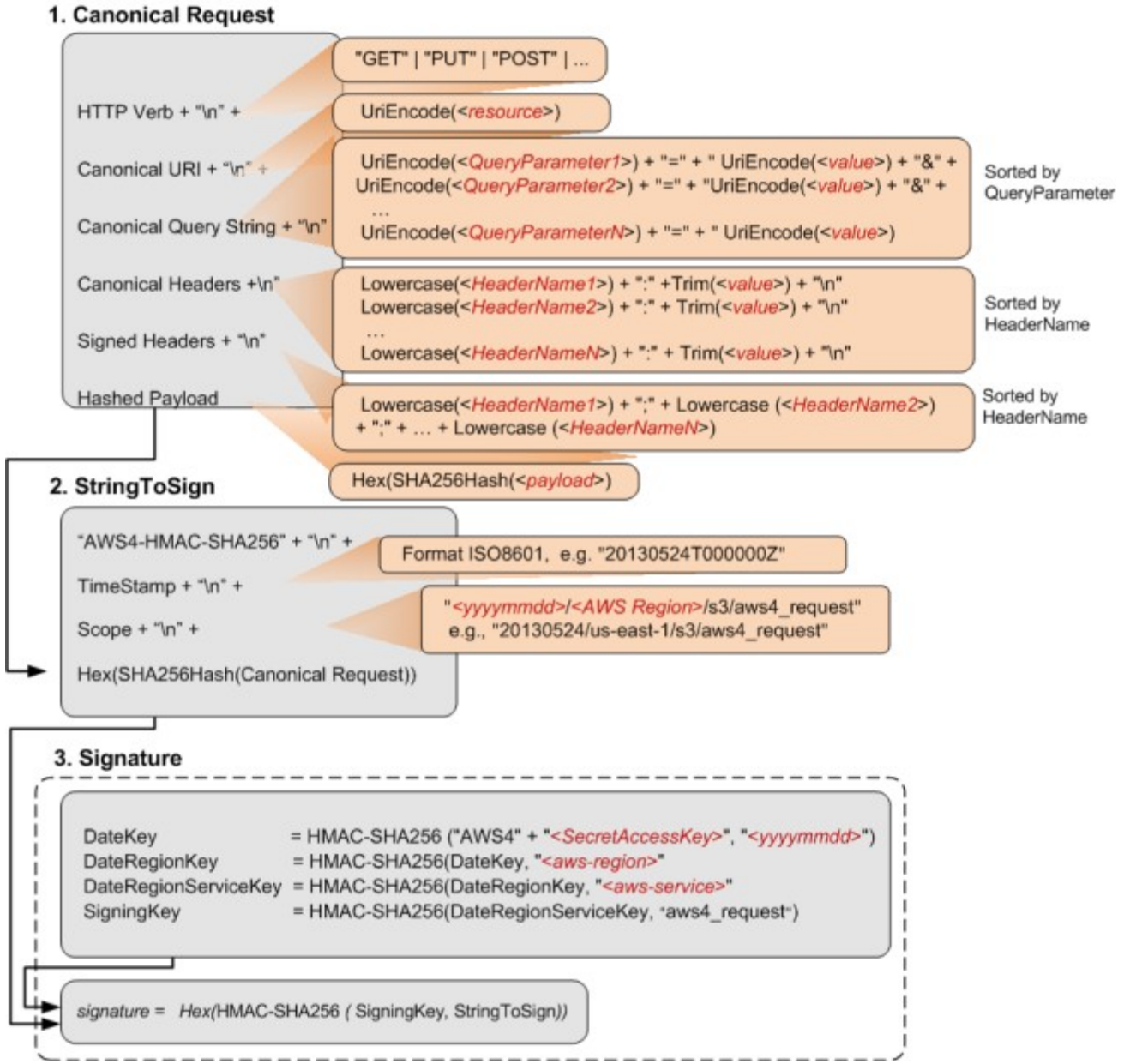
A native S3 interface for EOS/XrootD

Mano Ségransan, ID-SD Trainee

Supervisor: Andreas Joachim Peters

How it works

- Gateway between S3 clients and XrootD
- Implemented as an XrdHttpExtHandler
- Translates S3 HTTP requests into XrdPosix calls
- Authenticates request using AWS Signature V4
- Supports most upload and download operations



https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-header-based-auth.html

Challenges

- Store objects in a compatible way with EOS/XrootD filesystem hierarchy
- Some limitations:
  - Object name cannot end with a /
  - Object name cannot contain /./, /../ or only spaces between /
  - Object name cannot contain another objects' name as part of its name
- Examples of invalid names
  - foo/
  - foo/../bar, foo/./bar foo/ /bar
  - foo/bar/baz if an object named foo/bar already exists
  - foo/bar if an object named foo/bar/baz already exists

Supported Operations

- Create, delete, head and list buckets
- Put, delete, head, get and list objects
- Create, abort, complete and list multipart uploads

Limitations

- Only supports authenticating requests using the authorization header
- Does not support S3 Policies and ACL
- Authorization needs to be configured on EOS/XrootD

Example

- **File upload with awscli**
  - aws s3 cp file.txt s3://bucket/file.txt
  - Gateway will parse and verify the signature of the HTTP Put request
  - With the signature key it will then find the associated EOS/XrootD user
  - It will then try an XrdPosixOpen for writing on the corresponding file
  - XrootD/EOS will be responsible for accepting/denying this request depending on permissions
  - The gateway will then send the appropriate response to the awscli client

Further improvements

- More authentication methods: using query parameters, browser based POST uploads, streaming signed uploads
- Anonymous access for public content
- More S3 configuration options: lifecycle, versioning
- Encryption
- Different checksum algorithms

home.cern