SNIA®

Open-Source Multi-Vendor S3 Interoperability Tools: Ecosystem and Improvements

SNIA Cloud Object Storage Test Tools Technical Work Group

June 10, 2026

Table of Contents

Executive Summary

This SNIA white paper surveys the current landscape of available S3-based open-source API test tools, to assess their usefulness in driving multi-vendor S3 interoperability across diverse Cloud Object Storage (COS) deployment models (cloud, on-premises, hybrid, and multi-cloud). This paper furthers the goal of the COS Test Tools Technical Work Group (TWG) to foster more consistent customer solution experiences, since all global users require stable, accurate and reliable access to their data with reliable data management, regardless of S3 vendor implementation.

This S3 API Test Tool assessment helps implementation developers identify test methodologies, coverage, and feature gaps, as well as challenges to adapting available open-source tools to meet the needs of general multi-vendor interoperability testing. The conclusion proposes next steps and future initiatives to build industry consensus on developer best practice to promote wide-spread interoperability test automation.

Introduction

The ubiquity of object storage in hybrid-cloud infrastructures and strategies for multi-cloud integration has made cross vendor compatibility around object storage APIs critical for success. Motivated by member input, SNIA, an industry organization that develops global standards and delivers education on all technologies related to data, is driving collaboration on Cloud Object Storage (COS) multi-vendor interoperability, to help foster more consistent customer solution experience.

While there exist several heavily used object storage APIs, Amazon Simple Storage Service (S3) by Amazon Web Services (AWS) has been broadly adopted as the *de facto* standard API. SNIA members and their customers have expressed the need for seamless multi-vendor S3 interoperability, consistent with its position as a de facto standard. Until now, cloud object storage developers have worked on "S3 compatible" storage platforms independently, with no collaborative industry forum to help cultivate consensus on how to attain multi-vendor S3 interoperability between cloud object storage systems deployed in the cloud, on-premises, or hybrid and multi-cloud usage.

This gap is significant in contrast to multi-vendor collaboration in similar services such as SMB or NFS; SNIA members are now collaborating on S3.

COS Plugfests and COS Test Tools Technical Work Group

In response to this end-to-end heterogeneous need, SNIA has created two initiatives to foster industry collaboration and promote any-to-any multi-vendor interoperability: COS Plugfests and the COS Test Tools Technical Work Group (TWG). Providing support, the SNIA Cloud Storage Technologies (CST) Community has hosted several COS Plugfests since the SNIA Developer Conference (SDC) 2024, where participants found bugs, expedited inter-company interactions, shared best practices and knowledge, and tested interoperability. Under NDA between participating companies, participants discussed and identified common problem areas and brainstormed solutions to problems and gaps in tooling. Inspired by the success of the SNIA SMB3 Interoperability Lab, these Plugfests confirmed SNIA's role as a knowledge center and facilitator for vendor-neutral open-source collaboration.

This white paper surveys the current landscape of S3-based open-source API test tools and assesses their usefulness in the context of interoperability. This paper does not provide a ranking of any one test tool over another, but surveys test methodologies, coverage, and feature gaps. Furthermore, it will act as a foundation for future initiatives by describing next steps to build industry consensus on best practice to automate interoperability testing and enhance customer experience. For more information or to join the SNIA COS Test Tools effort, please contact us at costttwg-chair@snia.org.

Problems Facing S3 Object Storage Vendors

Many object storage systems claim to be "S3 compatible," yet there is no industry consensus on methods of verifying or comparing compatibility across vendors. Anyone asking the question "what is core S3 functionality" will likely be presented with widely differing answers, depending on the use-case and perspective of who is asking. Take for example defining support for even the simplest of operations, such as an object ingest. One must consider the interactions of additional headers and features, authentication and authorization, durability of the operation, or even which HTTP verb is being used. If an S3 compatible object store has not implemented certain features, such as conditional request headers or integrity checksums, ignoring a request header or providing an unexpected response may have serious consequences in data integrity.

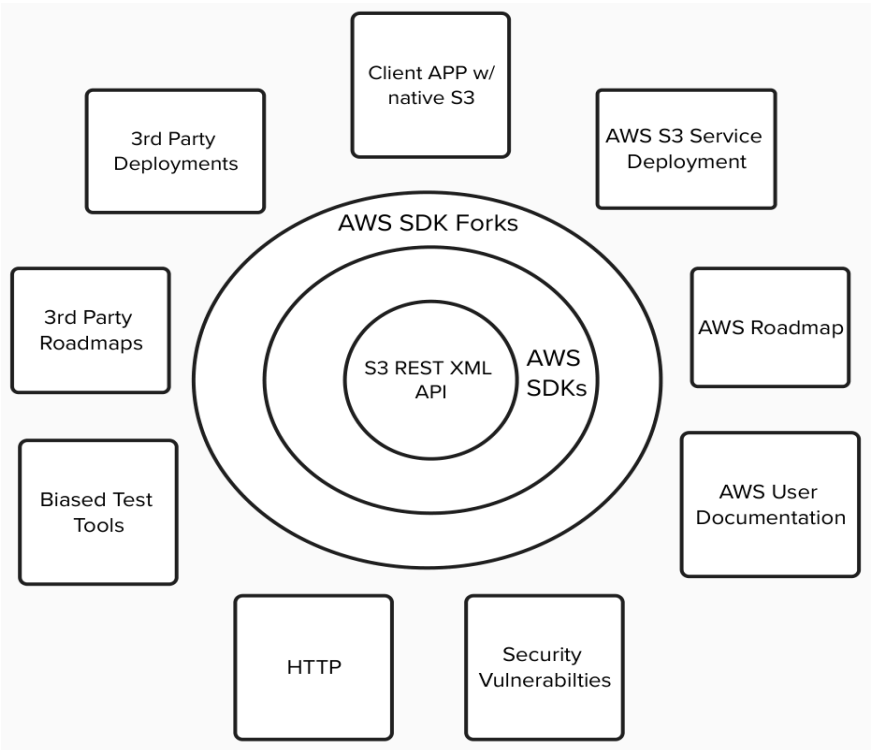

Diagram describing areas within the S3 API where interoperability issues can occur

Regarding multi-vendor compatibility, certain properties of the S3 ecosystem increase the likelihood of encountering interoperability issues, including the following:

- Lack of a detailed specification defining what the *de facto* core API is
- Continuous change to the S3 specification combined with lack of feature roadmap documenting new feature timing, and lack of versioning strategy with consistent backward compatibility methods
- Rapid change in AWS SDK usage patterns, along with new failure modes emerging as object storage technologies evolve
- Lack of functional testing following HTTP common practices to establish consistent basic multi-vendor interoperability, needed as a basis for more advanced workloads

While innovation around S3 has produced obvious benefits for the industry, third-party COS platform vendors and cloud service vendors have struggled to work around these issues. When interoperability issues occur, reliability suffers, and the ability to access and preserve critical data could be compromised. Detection and root cause of these types of issues can be hard and workarounds may be complex or impossible without coordination between stakeholders.

Additional challenges arise, as third-party vendors own and maintain their respective, independent software stacks, containing their unique features, capabilities, and extensions to the S3 API itself. These parties then create their own COS test tools, biased towards their own needs, or leverage available test tools created by others for different purposes, which may have similar but not identical needs and biases. Finally, third-party vendors lack an impartial, vendor-neutral industry forum to cultivate broad consensus on how to attain multi-vendor S3 interoperability.

What Should Interoperability Test Tools Do?

According to the SNIA dictionary, Storage Interoperability is defined as: The ability of storage devices, products, or systems to work together in a correct, predictable and interchangeable fashion.

At a high level, a generic, multi-vendor interoperability tool for S3 compatible object storage could be testing a server's implementation of the API, or a client's consumption of the API. Such a tool should broadly identify support for API operations, such as GetObject, PutObject and their variations in headers, which ultimately affect outcomes. Tools should also consider ancillary systems not strictly related to storage, namely authentication and authorization, and integration into other systems such as notification, audit, and logging systems.

For the purposes of this paper, we define "foundational S3 operations" as the following set of S3 APIs that perform primitive object storage operations. Each of these APIs has numerous variations that may affect behavior and outcomes, so when referred to in this context we mean support of any of the possible variations of each API.

FOUNDATIONAL S3 OPERATIONS

Creating or overwriting an object	PutObject
Retrieving an object	GetObject
Retrieving an object's metadata	HeadObject
Deleting an object	DeleteObject
Discovering objects	ListObjects

Overview and Assessment of Available Open-Source S3 Interoperability Tools

In this paper we examine four different open-source projects available on Github (Ceph S3 Tests, MinIO Mint, Snowflake S3Compat, and Versity S3 Gateway) that test S3 API functionality. While some tools may have performance related tests, we will only examine tests relating to interoperability. We note that some test tools are not presented as standalone tools for generic S3 implementors, rather they are tools designed to aid in the development for a specific S3 implementation or client.

FOUR OPEN-SOURCE PROJECTS AVAILABLE ON GITHUB

Ceph S3 Tests	Active Project	Of the four tools assessed, Ceph S3 Tests offer the most extensive coverage of the AWS S3 API, with STS and lifecycle/storage class, and ACL/IAM testing being a particular strength.
MinIO Mint	Still available as read-me, however, project Archived starting in March 2025.	Of the four tools assessed, MinIO Mint covers most of the AWS S3 API, using many of the AWS SDKs, MinIO s SDKs, and industry-standard command line tools. Additionally, Mint has the broadest coverage of both SDKs and clients, surpassing all other evaluated tools combined.
Snowflake S3Compat	Still available, however, last update in April 2025	Of the four tools assessed, Snowflake S3Compat covers a subset of core S3 API functionality, which is specific to the Snowflake product.
Versity S3 Gateway	Active Project	Of the four tools assessed, Versity S3 Gateway exposes an S3-compatible API which acts as an S3-compatible client. This helps support interoperability test coverage on both ends of an S3 connection.

When assessing test capabilities beyond core foundational S3 operations, the three more full featured tools (Ceph, Mint, and Versity S3 Gateway) all include tests for the following features to varying extents:

- Checksum Headers
- Conditional HTTP Requests
- Bucket ACL and CORS Policies
- Server-Side Encryption
- S3 Select
- Bucket and Object Tagging
- Versioning
- Object Lock

Ceph Interoperability Test Report

What is Ceph?

Ceph is an open-source storage platform that provides object, block, and file storage. Ceph began its life as a doctoral research project in 2004, was acquired by Red Hat in 2014, and has been a leader in open-source, software defined storage. Ceph's object storage is based on *the reliable autonomic distributed object store* (RADOS), which can be accessed by either the AWS S3 API or the OpenStack Swift interface. Ceph's product is widely used by cloud service providers, on-prem enterprise users, high-performance computing environments, and via on-demand cloud-computing in Kubernetes or OpenShift computing environments. Ceph implements most of the AWS S3 API and offers support for adjacent APIs such as AWS Secure Token Service (STS).

Approach to interoperability testing

As a fully open-source project, Ceph's tests are also available to anyone who wishes to inspect or utilize the code. Ceph's main repository contains an immense amount of testing, covering all components of the Ceph product, which includes testing the S3 API. These tests are executed using the Teuthology integration test framework, which automates and orchestrates execution of testing component suites on Ceph clusters. This is *not* what is under evaluation. The scope of this tool evaluation is on the Ceph S3 Test suite, maintained in a separate repository, which is specifically designed as a set of compatibility tests for S3-like APIs.

Ceph S3 test suite

Ceph S3 Tests are provided as an MIT licensed codebase designed by the Ceph team for validating S3 API compatibility. The suite is written in Python, and tests utilize the Boto3 S3 library. In the past tests were also implemented using Boto2 and some documentation still refers to the now removed test cases. A fork of the test suite prior to dropping Boto2 testing is maintained by the author of s3proxy. Standard Python testing frameworks (tox and pytest) are used and can be easily integrated into most standard CI/CD pipeline tools for automation. Configuration is done via a single file, which asks for information such as endpoint, credentials, TLS configuration, etc., however there is a lack of explanation for some of these options, which may lead to difficulty for use against non-Ceph implementations.

The Ceph team and their tests have been a foundational part of interoperability testing since the first SNIA COS Plugfest, September 2024, and we thank them for their contribution thus far.

Findings and Observations

Tests are organized into logical groupings of test suites (IAM, S3, S3 Select, STS, SNS, etc.) to enable focused test runs, as the repository contains hundreds of tests with broad coverage of the S3 and adjacent APIs. Of the tools assessed in this paper, Ceph S3 Tests offer the most extensive coverage of the AWS S3 API, with STS and lifecycle/storage class, and ACL/IAM testing being a particular strength. The tests are oriented at reproducing Ceph's specific implementation and users may encounter differences in behavior if comparing with other implementations. As an example, there are annotations such as "fails_on_aws" (or "unreliable") when there are explicit deviations or known issues with the test case.

The suite is under active development, with commits to the repository weekly. Unfortunately, the repository does not use git tags or Github releases, or versioning, making it difficult to assess improvements in Ceph or other implementations over time. Coverage of various aspects of the S3 API is always improving, but the tests aren't well documented or organized in a manner that makes it easy to identify what test cases are explicitly covering functionality. Documentation of each test's expectations and coverage would be valuable. We also believe that the output of the test could be improved with clarity around failures. Investigating the code can be daunting as each suite is implemented as a single file, the base S3 test suite being a single 20,000-line file as an example.

Test runtime is long and depends on network and storage throughput. Running the tests on a dedicated virtual machine in a cloud computing environment had tests completing successfully in under an hour, but with errors, execution time could easily double. Furthermore, many of the test cases that are marked as failing on Ceph (rgw) are Boto2 tests that have not been fully ported to Boto3. Unlike the greater Ceph project, contributions from non-RedHat or Ceph members are rarely accepted for S3 Tests and there are no contribution guidelines to follow. This is disappointing given its close relationship to a highly notable FOSS project.

Clients and SDKs used

- Boto3

Additional APIs and features tested

- AssumeRole (STS)
- Bucket Notifications
- Bucket Lifecyle

MinIO Interoperability Test Report

What is MinIO?

MinIO began its life in 2014 as an open-source, cloud-native implementation of S3-compatibile object storage. MinIO describes itself as “One of the earliest adopters of the S3 API (both V2 and V4) and focuses exclusively on S3.” A key to MinIO’s success has been the MinIO client software (mc), which provides a modern alternative to Unix filesystem commands along with easy management of storage. In addition, MinIO has developed several S3 compatible SDKs in various languages as an alternative to AWS S3 SDKs. Today, MinIO continues to develop object storage software under the AIStor branding while providing community editions of the client and object store server under the GNU AGPLv3 license. The SDKs and the tool Mint, which is the focus of this report, are available under Apache 2.0 licensing.

Approach to interoperability testing

MinIO’s stated goal is to “strictly follow the S3 API” and deliver a storage system that offers a complete set of all S3 API calls. In 2016, the MinIO team released, s3verify, a small test suite written in Go to validate AWS S3 v4 API requests. The team developing s3verify believed that dependencies on 3rd party SDKs could potentially mask or correct bugs in this area and developed s3verify from scratch as a tool for the greater object storage community. S3Verify was replaced by Mint in 2018, as a collection of correctness, benchmarking, and stress tests. Mint is container native and runs correctness tests using a broad mix of official AWS releases, MinIO developed tools, and 3rd party client implementations.

MinIO Mint test suite

MinIO Mint is still available as read-me, however, the project was archived, starting in March 2025. The Mint test suite provides a very useful reference, having wide-ranging test application which covers most of the AWS S3 API using many of the AWS SDKs, MinIO’s SDKs, and industry-standard command line tools. Out of all the tools evaluated, Mint has the broadest coverage of both SDKs and clients, surpassing all other evaluated tools combined. As a container-native application, Mint is easy to run using Podman or Docker. Pre-built images for x86-64 are provided and our examination of the Dockerfile and build scripts lead us to believe that compiling for other architectures could be done with little to no modifications. Configuration is done through environment variables, and the full suite can be executed with a single command. Test case results are stored in JSON files, and the

console log is relatively sparse, only showing the time it took to run each suite and which test failed if any.

During the February 2026 SNIA COS Plugfest, participants were encouraged to attempt to run Mint against participating storage providers, as we did for the Snowflake suite. Testers generally executed these tests via cloud computing resources resulting in runtimes between 30-60 minutes. In a subsequent on-prem evaluation using the latest "edge" image, the tests took approximately 2 hours to complete.

Findings and Observations

Test cases are grouped by SDK and programming language; coverage of the API can vary based on those variables. For example, there are 24 tests for the aws-sdk-go-v2 tests versus well over 100 tests for the minio-go SDK. This is something to be expected, as the MinIO SDK is their own piece of software and Mint's MinIO SDK test suites directly source the functional tests of their own implementations. Some storage developers might find it useful to have a consistent set of tests applied to each SDK test case, which may be possible through significant refactoring of the codebase. There is some variability in the reporting, some failures can be easily identified by the details in the JSON report, however others may require a little digging in the source code of the test or SDK. While it is possible to run the test suite for a single SDK or tool, running individual test cases is not possible for most of the suites. This "fail-fast" approach makes sense in CI/CD for MinIO development but may not be desirable for other COS developer implementations.

There is a discrepancy between the releases and tags shown on Github and the image tags available on Dockerhub. It was also noted that, at the time of writing, there were only two versions, the initial 2018 release, and a recent bump in March of 2025. This ambiguity results in not being able to easily determine which specific set of client SDKs are being used; users can easily modify the version number of tools, commit those changes, and rebuild the tool and re-run the tests with versions more relevant for their use-cases.

While Mint's origins in s3verify began with strict S3 API conformance as the focus, we believe the now archived MinIO Mint test suite represents a useful reference to help build future S3 interoperability test tools.

Clients and SDKs used

- AWS SDKs
  - aws-sdk-go-v2
  - aws-sdk-java-v2
  - aws-sdk-php
  - aws-sdk-ruby
- MinIO SDKs
  - minio-go
  - minio-java
  - minio-js
  - minio-py
- CLI Clients
  - awscli (Python3 botocore)
  - MinIO Client (mc)
  - s3cmd

Additional APIs and features tested

- Trailing headers
- Bucket Notifications
- Bucket Lifecyle
- Pre-signed URLs

Snowflake Interoperability Test Report

What is Snowflake?

Snowflake is an integrated data platform which connects a variety of unstructured, semi-structured, and structured data platforms with elastic compute and a variety of ways to connect, process, and interpret data. One of the foundational use-cases of Snowflake's platform is allowing users to directly connect to object storage via an S3-compatable API, allowing customers to utilize massive amounts of data in-place, while meeting complex security and compliance requirements.

Approach to interoperability testing

Snowflake S3Compat API Test Suite offers two methods of ensuring S3 API interoperability between various COS providers and its platform. For select cloud platforms, Snowflake offers an integrated storage validation command in their proprietary software which can be utilized by developers. Our focus, however, is the open-source Snowflake S3Compat API Test Suite, available on GitHub and distributed under an Apache 2.0 license. Snowflake provides a fairly well documented user guide

for utilizing S3-compatible storage and actively suggests using this tool for evaluating COS vendor interoperability with Snowflake products. Although Snowflake is a cloud-based product, they advertise support for on-premises S3-compatible storage and specifically suggest to customers and vendors to use this test suite. Snowflake states that S3 API support is primarily used for:

- Querying data from an external stage without loading the data into Snowflake
- Reading and processing unstructured data
- Copying files in S3-compatible storage from one location to another

These three use cases indicate a basic set of API operations, primarily focused on ingesting data from S3 into data processing stages.

Snowflake S3Compat API test suite

The last update for the Snowflake S3Compat API test suite was made in April 2025. This test suite provides a useful reference, providing two basic functions for anyone looking to ensure basic interoperability between Snowflake and an S3-compatible object store: tests for storage API operations and a basic set of performance test tools. The test suite is written in Java and utilizes the legacy AWS SDK for Java. It requires Java 1.8 or higher to run and uses Maven to build and execute tests. A GitHub account is required to access a fork of the spf4j-ui dependency, which slightly complicates using the tool. Basic instructions are provided in the README, and a basic knowledge of the Java ecosystem is useful if any issues are encountered during build or execution of the tests.

Differing from the other tools examined and owing to Snowflake's own product interoperability with storage that claims S3-compatibility, the suite does not perform setup such as the creation of buckets or ingestion of objects. Prior to running the suite, the user must supply two buckets, one accessible bucket and a second bucket that is inaccessible for the given credentials. The primary bucket must also be filled with at least 1001 objects for the paginated listing test to succeed. Test configuration consists of 9 environment variables and invocation by way of the "mvn test" command, a familiar workflow for Java developers. Either the entire suite or individual test cases may be run. Snowflake documents the set of S3 APIs and functionality that are tested in the suite, and at the time of writing, we found that this list was mostly accurate, with the only omission from the list being the ListObjectVersions API. The Snowflake suite is the only tool that we have examined that explicitly lists in documentation what APIs are tested whereas most tools mention general API concepts.

Snowflake provides a helpful troubleshooting section of their README, which can help a COS developer or Snowflake user understand specific failure types and their impact with overall

Snowflake use of S3-compatible APIs. Describing acceptable failure states or deviations from AWS S3 behavior is an incredibly useful inclusion.

Findings and Observations

Test runtimes were between 5-10 minutes. In a subsequent on-prem evaluation, the suite took about 5 minutes to complete. We did not run the performance benchmark, as it was out of scope for this white paper.

Application and use-case focused test suites, like this one, offer both end-users and storage providers a clear and effective means of testing exactly what is required for using their product. Snowflake's suite is concise, and developers with working knowledge of Java will be able to easily understand what each test case involves. There are several critiques that we have with this test suite. As mentioned earlier, the test requires pre-created buckets with one of them containing over 1000 objects. This setup would need to be performed using some other client tool or system, which does raise the level of user knowledge required, something that may hinder its usefulness as a "checklist" item for procurement and similar contexts. We believe that these tests should be self-contained and provision their own buckets and objects, and clean up after themselves, unless configured to do otherwise (for debug purposes). If a user doesn't complete the setup process or the tests fail during execution, we encountered scenarios that required manual intervention to correct environmental issues and rerun the tests. Ideally this would be handled by the suite, but even scripts to provision and clean up buckets would be useful. Logging was also a minor difficulty, as logs to the console typically consisted of a Java backtrace and possibly an S3 error response, but deeper investigation requires knowledge of configuring log4j and the underlying AWS SDK. Additionally, because each API is tested in multiple ways in a single test case, failures can be difficult to understand or misleading, and later sub-tests are not run at all.

During the February 2026 SNIA COS Plugfest, participants were encouraged to attempt to run the Snowflake test suite to gather additional coverage data for participants and gather diverse feedback about the tool itself. Participants found that HTTPS testing was difficult and integration with the Snowflake service was complicated and not always deterministic. The troubleshooting section however does allude to these issues and mentions the need for a "valid SSL certificate", which we interpret to mean a commercial Certificate Authority signed certificate is needed for proper compatibility with Snowflake's products. Some participants encountered issues with their development environments and the components of the Java ecosystem required to build and

execute the tests; a pull request (and a fork) exists that offers the capability to containerize the test suite, which resolves the Java environment issues.

We believe the Snowflake S3Compat API test suite represents a useful reference to help build future S3 interoperability test tools.

Clients and SDKs used

- AWS SDK for Java v1 (Legacy Java SDK)

APIs and features tested

Snowflake is an outlier in the tools assessed and lists each API method tested. We include the full list here to demonstrate the unique perspective of a storage consumer concerned with specific API methods.

- GetObject (Including pre-signed URLs)
- PutObject
- ListObjectsV2
- DeleteObject
- DeleteObjects
- GetObjectMetadata
- CopyObject
- GetBucketLocation
- ListObjectVersions

Versity Interoperability Test Report

What is Versity?

Versity offers a variety of large-scale data management products to create a data management platform to ingest and process a variety of structured and unstructured data from a variety of storage products and protocols. Versity's S3 Gateway (versitygw) serves as a bridge between file-based storage systems and applications that rely on S3. The software and tests are open source with an Apache 2 license and under active development in partnership with Los Alamos National Laboratory and the Pawsey Supercomputing Research Center.

Approach to interoperability testing

Versity's S3 Gateway test suite is intended for validating S3 API interoperability, ensuring that Versity's product can work with any storage provider that claims that their product is "S3 compatible." As the S3 Gateway software exposes an S3-compatible API and acts as an S3-compatible client, the testing suite is unique for being particularly concerned with interoperability on both ends of a S3 connection. The suite is designed and developed to have a minimal and modular dependency footprint, utilizing the REST API and common S3 CLI interfaces (awscli, s3cmd, or mc), all of which are optional. Tests are written using AWS's behavior as the source of truth for S3 operations; this may not conform with Versity's own software's behavior, which is valuable for matching AWS, but could be seen as an inflexibility to developers in the at-large object-storage community who might desire flexibility in API behavior. Tests cover a lot of the basic S3 object and bucket API, as Versity utilizes a comprehensive integration test suite for testing their client's behavior.

Versity Gateway test suite

In contrast with the CLI SDK based integration tests, the REST tests are assisted by a golang S3 client implementation that does not use an SDK to generate cURL commands with the appropriate headers and signatures. The testing framework used is the Bash Automated Test System (BATS) with setup routines and test cases contained in shell functions. Testing can be accomplished using the provided Docker Compose script or via a local install. Configuration is handled by both environment variables and text config files. The tests require two sets of AWS credentials, although one can be a dummy credential, and two bucket names. There are environment variables to configure options like preserving buckets and files, logging configuration, and the ability to skip certain test scenarios that test policy and ownership controls.

Findings and Observations

Code is maintained in the Versity S3 Gateway repository with an Apache 2.0 license and is under active development. A notable differentiator from other tools is the implicit versioning that the test suite gains by existing within a release-versioned repository. Not only does the versioning allow the test suite to have a 1-1 compatibility mapping with Versity S3 Gateway, but it also allows other COS developers to clearly communicate their own S3 implementations test results.

Test cases in each suite are for the most part self-contained, handling setup and teardown as necessary. This allows the rest of the suite to continue running even if a prior test case fails, a

behavior which is useful when testing other S3 implementations. The option to preserve bucket contents or to not delete and re-create buckets may be useful in situations where a COS implementation may have limitations or for AWS accounts without full permissions.

Suites can be run individually, or several can be run in a back-to-back manner. While there is an option to run all the tests, the tool does not recommend it. Logging for the tests is basic, with a trace from the offending failures printed directly to the main log file. Valuable debugging information is present, in terms of what failed and when. Of note, the REST test logs include the sequence of cURL requests sent, and most of the other test cases can be configured to keep the request body or responses as a file for inspection.

One hiccup encountered during testing is that the default Docker configuration is set up for arm64. There were also some minor issues with setting up a local install relating to using a packaged BATS installation. We were unable to run the s3cmd tests due to Python incompatibility with newer Linux distributions. These problems occurred when deviating from the documentation to adapt to different environments and illustrate the value of providing container images for test tools.

At this time, IAM (referred to as users tests) and ACL test scenarios only work with Versity, both of which are easily skipped in the configuration. These scenarios may be run on non-Versity S3 implementations by modifying the relevant setup and teardown routines, a task that is relatively straightforward given the functional shell code structure.

Clients and SDKs used

- S3 REST API
- awscli (Python3 botocore)
- MinIO Client (mc)
- s3cmd

Additional APIs and features tested

- Trailing headers
- Chunked Uploads
- Pre-signed URLs

Common Limitations and Questions

At a high level, our assessment found that there is no one available COS test tool that is ready to fully satisfy general interoperability needs across a wide spectrum of third-party vendors. The tendency for open-source test tools to favor the needs of their originating projects limit coverage and suitability to directly represent broad industry consensus. However, the above assessment provides a wealth of insight on how existing practices may be extended, which will be covered in future work.

Handling varying extents of support for S3 APIs

A multi-vendor S3 API compatibility test tool should have, at the minimum, a stated expectation for differences in behaviors and a method to manage these differences. Without this, comprehensive testing can be a difficult proposition, as each additional feature under test may have to consider interactions with other features leading to complicated test behavior and increased runtime. For example, testing access controls with bucket policies may involve multiple levels of identity management support. A basic S3 implementation might use local user accounts, whereas advanced implementations may integrate with a multi-tenant SSO identity federation or IAM service.

Negative test cases

We recognize the difficulty found in generating vendor-neutral error responses from a storage system, which is why most open-source test suites only validate responses in successful cases. For example, 2 implementations could respond with HTTP status codes 400 and 422 for the same invalid operation.

Integration with auxiliary and adjacent APIs and services

Another difficulty faced by test developers is defining where the S3 API begins and ends, for example, bucket and object policies, tiering, billing, authorization and authentication are all components that may or may not be “core” to the S3 object storage API and can be incredibly difficult to test. Authentication and authorization are areas where S3 implementations differ significantly and may involve tertiary systems and protocols completely foreign from AWS’ implementation.

Narrow focus on certain clients

Some available open-source tests may make what we call biased choices on testing approaches. For example, many choose to utilize a specific AWS SDK for their preferred language (such as boto3 for Python) as a foundational component. Tests may attempt to check behavior across different SDKs, indicating differences in behavior as the backing SDKs change; however, no tool exists to directly compare multiple SDKs for the same API call. Some open-source developers have created suites that operate via the REST API directly, carefully crafting requests to isolate themselves from any hidden behaviors contained within SDKs, CLIs, or other intermediate software that may obscure subtle differences between storage systems. This can be useful for developers trying to implement their own S3-like storage system but could be difficult to implement for more complicated portions of the S3 API. SDKs offer ease of use when compared to the primitive REST API. In testing this becomes a major tradeoff as SDKs commonly normalize or correct inputs for user convenience, which may lead to developers expecting to test certain conditions, yet inadvertently not implementing their desired test case. HTTP wire traces may catch such errors at the cost of significant log volumes.

Broader community engagement

We found that the available open-source test tools were often oriented towards storage developers who develop or work with S3-compatible storage and exist to serve the purposes of their own product. Most are not necessarily intended for a broader audience or to be used with other implementations. A challenge we observed is encouraging existing open-source projects that include S3 test tools to accommodate the needs of the broader community's implementations of the S3 API.

The Road Ahead

The S3-compatible API ecosystem is diverse and robust. It has yielded multiple, production-ready object storage solutions, a variety of SDKs and software utilities. Some of these ecosystem contributions aim to serve a specific purpose and are released to the public as a benefit to those who might have similar goals or may provide a foundation for others to build similar tools. We applaud the efforts of these organizations and developers in building the tools we examined. This white paper is a call to action for the object storage community to assist in improving these tools or addressing gaps that may exist.

We set out to survey as much open-source software, attempting to address S3 API interoperability issues. Unfortunately, that means we couldn't test every open-source suite that was available. New software is always emerging, like the Datadobi S3Test suite, which was recently released, or S3 Tester, by MagaluCloud, which wasn't on our radar until recently. We suspect that creating tests to maintain interoperability exist (at some level) for every different S3-like storage implementation. We also recognize the complications that might prevent these tools from being released as open-source software. Complicating things further, we acknowledge that most open-source software is released primarily for the benefit of those developing it.

Since SNIA COS interoperability initiative provides a valuable multi-vendor expert gathering in the industry, it is our belief that COS Plugfests and the COS Test Tools TWG are uniquely positioned to facilitate the development of novel open-source, vendor-neutral tools to advance interoperability efforts. Broad multi-vendor collaboration is necessary to achieve industry consolidation of consistent best practice and consensus-based lessons learned, rather than additional proliferation of numerous stand-alone interoperability tools. SNIA is recognized and trusted for its leadership, standards, and expertise; SNIA TWGs share a common goal to advance and standardize both established and emerging technologies in a vendor-neutral environment.

We hope that AWS will participate in this effort, as they have recently expanded open-source resources for S3 developers via Smithy, a new protocol-agnostic interface definition language, along with the release of Smithy models of AWS services, including S3. This, combined with our exploration of existing open-source S3 test tools, leads us to believe that this could strongly influence, or become foundational in creating tools that are useful for maintaining basic S3 API compatibility and maintaining the flexibility required across different COS solutions.

In conclusion, we propose exploration to build open-source multi-vendor COS test tools that are specifically targeted at basic compatibility with the AWS S3 API, as previously referenced, keeping in mind the need for flexibility along with the following attributes:

- Suitability to promote and cultivate broad industry consensus and effectively attain multi-vendor S3 interoperability via a vendor-neutral industry forum
- Specification driven multi-vendor COS open-source test tool architecture that helps identify and accommodate differences between vendor behavior
- Define foundational common API components across key interoperability workload profiles, including cloud, on-prem, hybrid, or multi-cloud storage deployments
- Extend existing practices to reduce development and support costs, as well as create better S3 compatibility outcomes for both consumers and storage developers

Once implemented, such tools could build a strong foundation of common API components allowing deeper focus on difficult or rapidly evolving areas of interoperability. The SNIA COS Test Tools TWG will continue to investigate other tools, such as: a set of open source COS performance test tools, tools to test interoperability between different COA APIs (e.g., Azure to S3, GCS to S3, and OpenStack Swift to S3), and interoperability with ISO standard SNIA Cloud Data Management Interface (CDMI™).

Lastly, although the S3 compatible object storage industry is large, those who utilize S3 are far more numerous than storage developers, which is reflected by the number of SDKs, utilities, and resources that exist for the global user community. Bottom line, all global users require consistent, stable, accurate and reliable access to their data with reliable data management, regardless of S3 vendor implementation.

For more information or to join the SNIA COS Test Tools TWG, please contact us at costttwg-chair@snia.org.

About SNIA

About SNIA

SNIA is a not-for-profit global organization made up of corporations, universities, startups, and individuals. The members collaborate to develop and promote vendor-neutral architectures, standards, and education for management, movement, and security for technologies related to handling and optimizing data. SNIA focuses on the transport, storage, acceleration, format, protection, and optimization of infrastructure for data.

SNIA
5201 Great America Parkway, Suite 320, Santa Clara, CA, 95054
Phone:719-694-1380 • Fax: 719-694-1385 • www.snia.org