Single Namespace Requirements

Table of Contents

[**1. Introduction: 2**](#_heading=)

[1.1. 2](#_heading=h.t3atrv4mytrv)

[1.2. Genentech / Roche 2](#_heading=h.qcrt38r9ks18)

[2. Purpose 2](#_heading=h.30j0zll)

[2.1. 2](#_heading=h.teakh6irdtdf)

[2.2. 2](#_heading=h.1uq18hgyh20m)

[3. Audience 3](#_heading=h.1fob9te)

[4. Scope 3](#_heading=h.3znysh7)

[5. Definitions 3](#_heading=h.2et92p0)

[6. Acronyms 4](#_heading=h.tyjcwt)

[7. Project Description 4](#_heading=h.3dy6vkm)

[7.1. 4](#_heading=h.d1ocf6mp2od7)

[7.1.1. User Stories - Guardant Health 4](#_heading=h.dnrkr0h8ku34)

[7.1.2. Requirements 5](#_heading=h.ahoyd3mnvto8)

[7.2. Genentech / Roche 5](#_heading=h.ay3y72xajpre)

[7.2.1. Use Cases 5](#_heading=h.6cqrhe1foky4)

[7.2.2. Requirements 5](#_heading=h.j9vzxphztt0s)

[8. Common Single Namespace 5](#_heading=h.hjvn327ilrag)

[8.1. Namespace Requirements 5](#_heading=h.bp4wg5fvg4z7)

[8.2. File Storage Requirements 5](#_heading=h.by7b23z5rowd)

[8.3. Object Storage Requirements 6](#_heading=h.ylvwethxxkzh)

[8.4. Interactions Between File and Object Storage 6](#_heading=h.9pkv4r6llube)

[8.5. Security 7](#_heading=h.js8y5ha0dh0b)

[8.6. Interoperability 7](#_heading=h.xcituy5a7hue)

[8.7. Non-core Requirements 7](#_heading=h.5zzxg3mu9wcm)

[9. Conclusion 9](#_heading=h.44sinio)

[Version History 9](#_heading=h.2jxsxqh)

# Introduction:

## Guardant Health

Guardant Health is a precision medicine company focused on transforming patient care and conquering cancer using data. They produce liquid biopsy tests to detect cancer by analyzing blood samples. Their Shield test, recently approved by the FDA and covered by Medicare, has generated 70 Petabytes (PB) of data from 500,000 patients. This data is expected to grow to 700PB per year, with 80% of this data remaining “hot,” meaning it is frequently accessed.

Guardant Health is undertaking a data management initiative to address the challenges of managing large datasets in a single namespace. Data access, movement, and management is a ubiquitous problem across many industries. This is true for any industry with moderate amounts of data and multiple tiers of storage, remote locations for data processing, hybrid or multi-cloud services. Moving data between locations is a problem. Keeping data coherent between data islands is difficult. Data archiving and retrieval are challenging. Disparity of access types can confound and delay development.

Their aim is to have universal data access across all computational locations, encompassing on-premises and remote high-performance computing (HPC) systems, as well as cloud-hosted services.

## Genentech / Roche

is a Swiss multinational healthcare company with two main divisions, pharmaceuticals and diagnostics.

Roche has over 100’000 employees worldwide and operates various sites on all continents with multiple data centers. It operates storage systems of all major vendors storing several 100PBs on-prem and similar amounts across the major cloud providers.

# Purpose

## Guardant Health

The purpose of the single namespace is to provide universal data access across all Guardant computational locations. This includes all on-premise data processing capabilities, remotely hosted capabilities, and cloud-hosted services.

This document is intended to spell out the requirements for the single namespace from the Guardant Health use cases. This will provide the necessary use case requirements for the design and implementation of the single namespace.

## Genentech / Roche

TODO

# Audience

This document is intended for the internal consumption of Guardant Health, its employees, and the members of the single namespace working group. These working group members are to use the design document as a baseline guide or seed for their own implementations of the single namespace and the development of interoperability standards between vendors.

# Scope

The scope of this document is limited to the requirements as seen from the perspective of Guardant Health’s use case. This is not intended to be a complete and comprehensive design document for all use cases of a single namespace in the general use sense.

# Definitions

* **Single Namespace (SNS)**: A unified namespace that provides a consistent view of data across multiple storage systems and locations.
* **Global Namespace**: A unified namespace that spans across different geographical locations and storage systems.
* **Cluster File System**: A file system that allows multiple nodes to access and manage files in a coordinated manner.
* **POSIX**: Portable Operating System Interface, a family of standards specified by the IEEE for maintaining compatibility between operating systems.
* **HPC**: High-Performance Computing, the use of supercomputers and parallel processing techniques for solving complex computational problems.
* **RBAC**: Role-Based Access Control, a method of regulating access to computer or network resources based on the roles of individual users within an organization.
* **API**: Application Programming Interface, a set of tools and protocols for building and interacting with software applications.
* **SDK**: Software Development Kit, a collection of software development tools in one installable package.
* **NFS**: Network File System, a distributed file system protocol that allows a user on a client computer to access files over a network.
* **SMB**: Server Message Block, a network file sharing protocol that allows applications on a computer to read and write to files and request services from server programs in a computer network.
* **S3**: Simple Storage Service, an object storage service that offers industry-leading scalability, data availability, security, and performance.
* IAM: Identity and Access Management: is a framework of policies and technologies to ensure that the right users have the appropriate access to technology resources.
* HA: High-availability: is a characteristic of a system that aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

# Acronyms

* **SNS**: Single Namespace
* **HPC**: High-Performance Computing
* **POSIX**: Portable Operating System Interface
* **PB**: Petabyte
* **RBAC**: Role-Based Access Control
* **API**: Application Programming Interface
* **SDK**: Software Development Kit
* **NFS**: Network File System
* **SMB**: Server Message Block
* **S3**: Simple Storage Service
* **CSI**: Container Storage Interface

# Project Description

The single namespace at Guardant Health will share all data from remote repositories to any client computational locales. These locales can be on-premises HPC systems, remote HPC environments, and any cloud service providers.

## Guardant Health

## User Stories

As a researcher, I want to access data from any location, regardless of its physical storage, so I can seamlessly collaborate with colleagues and analyze data in real-time.

As a system administrator, I want a unified view of all data assets, regardless of storage type or location, to simplify management and monitoring tasks.

As a data scientist, I need low-latency access to large datasets for HPC workloads, so I can accelerate research.

As a legal and compliance officer, I want to ensure that the storage solution meets regulatory requirements for data security, privacy, and immutability

## Requirements

Guardant Health’s single namespace vision is driven by several key requirements and concerns described below.

## Genentech / Roche

## Use Cases

Data flows through different stages, from acquisition (e.g. measurement device, computational pipelines, …) through different stages until reaching point of deletion.

TODO

## Requirements

TODO

# Common Single Namespace

## Namespace Requirements

* 1. **Single Namespace:** Establish a global, unified single namespace encompassing on-premises, hybrid, and multi-cloud environments.
  2. **Support for File and Object Storage**: The namespace should present itself as either file or object, seamlessly integrating them both.
  3. **Hybrid Multi-cloud Integration**: Support for moving data between on-premises and cloud environments.

## File Storage Requirements

* 1. **Frontend Access:** The file storage serves as the front end of the SNS, providing access to all data within the SNS.
  2. **POSIX Compliance**: The file storage must support POSIX standards to ensure compatibility with HPC applications.
  3. **Protocol Support:** Compatibility with existing HPC environments mandates support for NFS and SMB protocols
  4. **Container Storage Interface Support:** Current HPC environments use Kubernetes for services in the HPC workloads. The Kubernetes instances access the front end via the container storage interface. Support for CSI is necessary.
  5. **High Performance:** File storage should only be used as cache, optimized for high throughput and low latency to support HPC workloads. It is not a source of truth.
  6. **Clustered File System:** The file storage should be a cluster file system to share data across all nodes when interacting with the local cache of data.

## Object Storage Requirements

* 1. **Backend Source:** The object storage acts as the back end of the SNS and must be the source of truth for the data.
  2. **Protocol Support:** The S3 API should be the primary interface for object storage, providing comprehensive compatibility with industry standard tools.
  3. **Data Management**: The object storage must have data management capabilities, to enable seamless data movement across different locations.
  4. **Metadata Management**: The object storage should store all system and custom metadata, enabling comprehensive search capabilities.
  5. **Scalability:** The object storage, as the source of truth, needs to scale to hundreds of PB.
  6. **Durability:** The object storage, as the source of truth, needs to have high durability and data protection through erasure coding and replication
  7. **Data Integrity**: The object storage, as the source of truth, needs to ensure long-term integrity of data and guard against bit rot.
  8. **Compliance**: Support object versioning and object lock features to ensure compliance.
  9. **Adjustable Consistency**: Offer multiple consistency levels such as strong consistency that would ensure that any read or list request immediately reflects the latest write.

## Interactions Between File and Object Storage

* 1. **Data Tiering:** Support automated data tiering between the file storage and the object storage tier based on access patterns. Tiering should be policy-based allowing customization of tiering rules and schedules.
  2. **Data Hydration:** Data shall be pulled from the object storage to the file storage where it will act as a cache when requested by an HPC application.
  3. **Data Dehydration:** Data not used for a set period shall be dehydrated, leaving stubs pointing to the original object storage.
  4. **Strong Consistency:** There should be strong consistency between file and object, meaning that once a write operation is acknowledged, all subsequent reads must reflect the latest data.
  5. **File Stubbing:** When the data is tiered, the original file is replaced with a stub. The stub contains metadata pointing to the actual data in the object storage. Accessing a stub file should trigger retrieval of the data from the object storage transparently.
  6. **Metadata Transfer:** Additional metadata in the file storage should be transferred and preserved in the object storage in a standardized format.
  7. **Metadata Updates:** Updates to the object storage shall be reflected in the metadata and stubs in the file storage front ends.
  8. **Data Updates:** Only updated or changed data shall be pushed to the object storage during a dehydration event to reduce traffic.
  9. **Multi-backends:** The file storage must interact with multiple object storage backends, presenting them as a single POSIX file system.
  10. **Redundant-backends:** The file storage system must support a single stub pointing to multiple object storage backends for the same file, enabling redundancy, mirroring, and prioritization.
  11. **Reconciliation between backends:** When using multiple backends, reconciliation and consistency must be maintained by policy.
  12. **Resilient Operation:** Connectivity between the file and object storage may be interrupted. Continued operation is required even when connectivity is interrupted. Recovery and reconciliation is required once connectivity is reestablished.
  13. Multi-Protocol Access: The S3 buckets and objects can be accessed directly by users, not only by File Storage Layer tiering to the Object storage layer. Mechanisms should be in place to ensure eventual consistency between File and Object versions.

## Security

* 1. **Encryption:** Data should be encrypted both at rest and in transit for data protection purposes.
  2. **Authentication and Authorization:** Provide secure authentication and authorization mechanisms to verify user identities and grant appropriate access privileges.
  3. **Role-Based Access Controls:** The solution should provide fine-grained access control mechanisms to restrict access to data based on user roles and permissions. This includes the ability to restrict or grant access to remote sites.
  4. Identity and Access Management integration: Information for RBAC (e.g. access groups and member information) should be consumed from company managed IAM systems. Storage systems can cache this information for a specified duration to improve performance.

## Interoperability

* 1. **Cross Compatibility:** Ensure the solution can ingest data from and write to standard formats used by other vendor technologies, facilitating seamless integration.
  2. **Cross Compatibility Verification:** Conduct thorough testing to validate the solution’s compatibility with other vendor technologies.
  3. Cross Company Interoperability: Storage systems connected to a single namespace shall implement multi-tenancy in a way that proper access controls can be preserved while limiting sharing of only necessary user/group information.

## Non-core Requirements

There are several requirements not required for the minimum functionality needed for a single namespace for Guardant Health. These are nontrivial to accomplish but would be fantastic to have.

* 1. **Locking:** Implementing locking mechanisms for remote objects by local caches would be highly useful. This feature would allow for wider area collaboration without issues of last writer wins scenarios or potential data corruption.
  2. **Hints**: Hints would enable more sophisticated interactions between the file and object layer.
  3. **Extended File Metadata**: Beyond standard POSIX metadata, the ability to add additional metadata would be desirable.
  4. **Alternate Storage Technologies**:While not necessary for Guardant today, an alternate backend storage than objects would be a good goal for future flexibility.
  5. **Support for manifests**: Support manifests for more efficient communication and data retrieval. Manifests can help in organizing and managing large datasets more effectively.
  6. Action/Event Trigger: Upon reaching a certain condition (e.g. file created with a defined pattern), a specified callback method is invoked.

# Conclusion

Guardant Health is leading precision medicine by using extensive data to transform cancer care. Their FDA-approved Shield test has accelerated data growth, posing significant management challenges. To address this, they plan to create a single namespace for universal data access across all computational locations, including on-premises, remote HPC systems, and cloud services. This product requirements document outlines the high-level requirements of this namespace. It is intended for Guardant Health employees and the single namespace working group to develop interoperability standards and ensure seamless platform integration. While tailored to Guardant Health, the principles may benefit similar initiatives in other organizations.

For Genentech / Roche, access to data across divisions is a key factor for success. Facilitating access to data is getting even more central

# Version History

| Version | Date | Document Version History |
| --- | --- | --- |
| V.Alpha | January 7, 2025 | Initial draft of the document from William |
| V2 | January 20, 2025 | Revised version by Jonathan |
| V3 | January 22, 2025 | Revised version by William |