GFS Replication Guide

Published 2026-10-07

For the latest version of this document, see docs.hammerspace.com/home/downloads

HAMMERSPACE

Table of Contents

About This Documentation 1
1. Goal of This Document 3
2. Hammerspace Multi-Site Fundamentals 4
2.1. Key Features 4
2.2. Key Components 5
2.3. Recommended Use Cases 6
2.4. Unsupported Use Cases and Configurations 6
2.5. Data Orchestration Replication Flow Example - Staged Workflow 7
3. Prerequisites for GFS Deployment 8
3.1. Hardware Resources 8
3.2. Network Connectivity 8
3.3. Shared Bucket 8
3.4. Global NFS Shares 9
3.5. Global SMB Shares 9
3.6. Global S3 Servers/Buckets 9
4. Set Up Replication 10
4.1. Step 1: Prepare the Infrastructure 10
4.2. Step 2: Configure a Shared Bucket 10
4.2.1. Add a Shared Bucket Using the GUI 10
4.2.2. Add a Shared Bucket Using the CLI 13
4.3. Step 3: Create a Global Share and Enable Replication 14
4.3.1. Create a Global Share Using the GUI 14
4.3.2. Create a Global Share Using the CLI 20
4.4. Step 4: Define the Data Path with Objectives 21
4.4.1. A. Core Replication Objective 21
4.4.2. B. Local instance/Performance Objectives 21
4.5. Step 5: Implement Durability and Consistency Controls 22
5. Key Configuration Parameters 23
5.1. Manage Data with Objectives 23
5.1.1. "LogXfer" Objective 23
5.1.2. "Keep on cloud" Objective 23
5.1.3. "Keep on site X" Objective 23
5.1.4. Asynchronous Reservation tag-Based Objective 23
5.2. Snapshot Strategy 24
5.3. File Versioning and Undelete 24
5.4. Data Durability & Availability 24
6. GFS Best Practices and Guardrails 25
6.1. Addressing Specific Protocol Challenges 25
6.2. Handling Replication Delays and Disconnections 25
6.3. Metadata and Data Collisions 26
6.4. Handling Conflicts 26
6.4.1. File Conflict Scenario #1: Creating the Same Filename Across Multiple Sites at the Same Time 27
6.4.2. File Conflict Scenario #2: Writing to an Existing File on Multiple Sites at the Same Time 28

6.5. User Education for Workflows . . . 29
7. Monitoring and Alerting 30
8. Conclusion 31
Appendices **32**
Appendix A: Additional Resources 33
Appendix B: Appendix a - Replication Examples 34
B.1. Example #1 . . . 34
B.2. Example #2 . . . 34
Appendix C: Appendix B - Common Errors 37
C.1. Gfs-participant-add: Fails to Add a Site . . . 37
C.2. Gfs-participant-add: SITE-B Is Already Participant 1 of Share Home . . . 37
C.3. Error Message: Bad Credentials . . . 37
C.4. Share-replication-create: Remote Site Error: An Entity (Share) Named 'Home' Already Exists . . . 38

About This Documentation

The Hammerspace Global File System (GFS) enables a single logical file system to span multiple locations regardless of geographic distance. This allows for visibility and accessibility of Enterprise data to remote sites, regardless of where the data is physically stored, while adhering to access controls and permissions. The goal is to make data a **global resource** for your Enterprise, instead of files being confined to a single location.

This is an important distinction, since traditional storage limits data to being a local resource and the only way these systems can provide data accessibility to remote locations is by making copies of files. This leads to copy sprawl, significantly increases management complexity, and requires the manual staging and destaging of data.

Metadata and Data Disaggregation

Hammerspace has disaggregated metadata and data within our file system. In doing so, we are able to replicate metadata, the presentation layer of the file system, across multiple locations, making data visible and accessible regardless of where it is physically stored. Logically, it is a single system that is unbounded by distance.

The Hammerspace Anvil metadata servers communicate with one another, providing updates on changes that occur within the GFS every five seconds. Data is moved and/or replicated by policy or on-demand leveraging Hammerspace Data Orchestration.

Data Orchestration

Hammerspace provides **Data Orchestration**, a powerful capability that efficiently makes data available to remote locations. It leverages Objectives, a policy-driven, file-granular methodology.

The Hammerspace Anvil metadata server is our control plane. We orchestrate data through metadata. You can set policies for data locality based on any file metadata attribute including file name, file type, file size, creation date, modified date, last access, etc. You can also customize metadata and can organize and group files together based on your business requirements for specific projects, use cases and workflows.

Hammerspace replicates **instances,** not copies. When you copy a file, it is now an entirely different file with the exact same content. An instance is a file that can physically be in two or more locations but to the file system, it is the same file. This is the essential distinction.

Hammerspace uses an eventually consistent mechanism to ensure that all of the instances are aligned. We purposefully did not use global file locking because by its nature, this method over long distances can slow down performance to the point of being unusable. Additionally, if any of the sites participating in the global file lock become unavailable, then all of the data goes into read-only mode, which fundamentally can greatly impact productivity. The Hammerspace eventual consistent methodology does not impede performance, nor is it affected if a site becomes unavailable.

With Hammerspace, file attributes (name, size, permissions, modification times) and custom metadata alike are continuously updated between all sites ensuring that every user across the GFS sees the same file system structure. The data path includes the file content that is replicated asynchronously via a shared Object Storage system. File data moves only when ordered to by a Place-On Objective (proactively) or when a remote user attempts to access the file (on-demand).

 © 2026 - Hammerspace

Workflow Considerations

The Hammerpace GFS is unique and powerful. As such, some of the concepts and requirements are vital to understand in order to optimally leverage it. It is important to establish workflows that take into consideration its eventually consistent nature.

The use cases include distributed workflows and burst-to-compute across Enterprise data centers, remote sites, edge and Cloud.

This guide discusses this two-path architecture and outlines the best practices to ensure data is placed in alignment with the needs of the workflow including data durability, maximizing performance, and/or enabling workload locality.

© 2026 - Hammerspace

Chapter 1. Goal of This Document

The purpose of this document is to understand the core concepts, outline the setup, configuration, and management of eventually consistent remote replication.

This document will provide an overview of:

- Prerequisites and initial setup
- Global File System configuration and best practices
- Monitoring and Troubleshooting
- Use cases and situations that are applicable & those that may invoke risk

To open a support case, use the Hammerspace Support Hub. To download Hammerspace software and licenses, use the Hammerspace Licensing and Delivery Portal.

> Do not install additional or third-party software, agents, or packages directly on Hammerspace Anvil (metadata) or DSX (data) nodes. These nodes run a managed appliance software stack; anything installed outside that stack is unsupported and is removed during a software upgrade. Such software can also interfere with node operation. If you need monitoring or integration with an external tool, contact Hammerspace Support for a supported approach.

© 2026 - Hammerspace

Chapter 2. Hammerspace Multi-Site Fundamentals

Hammerspace is designed to provide an **active-active, eventually consistent Global File System (GFS)** across multiple geographically dispersed sites. It operates by replicating metadata and employs Hammerspace objectives to instantiate data at designated site participants, ensuring accessibility and resilience.

- **Eventually Consistent**: Changes made on one site will eventually propagate to all other replicated sites.
- **Active-Active**: All sites can be enabled to serve data and accept writes. Keep in mind that global accessibility can allow for active-active write patterns and requires consideration and management.
- **Metadata Replication**: Checkpoints of metadata are taken at regular intervals (defaulting to 5 seconds per share/export) to replicate metadata diffs so that the GFS is consistent.
- **Data Replication**: Data movement leverages a shared object storage bucket. Data is uploaded to the shared bucket by the owning site and pulled by requesting sites, either by objectives or on-demand.

2.1. Key Features

The Hammerspace Global File System offers the following key features:

- Active-active data access.
- Global access to data across multiple standard protocols (NFS 3, NFS 4.1, NFS 4.2, SMB 2.x/3.x, and S3). No proprietary client software is required.
- Custom metadata, which can be manually or automatically applied.
- Objective-based data policies, triggered by any combination of file system and/or custom metadata to automate a wide range of data services and data placement actions.
- Optimized and transparent data orchestration across sites with integrated global deduplication and compression.
- Secure connections and encryption of the data payload during transfer and at rest in cloud/object storage.
- Global snapshots that can be stored anywhere.
- Global undelete.
- File-granular WORM, which can be part of an automated objective triggered by any metadata fragment.
- Anti-virus scanning.
- Access-based Enumeration (ABE) across NFS, SMB, and S3.
- Conflict resolution with cross-site ownership versioning.
- Per-site data management for optimal resource usage.
- Heterogenous storage support. Each site can use one or more storage platforms of any type, from any vendor.
- Capacity agnostic. Only requires enough capacity for active data.

 © 2026 - Hammerspace

- Intelligent caching. Drive local data availability through automated objectives triggered by any combination of multiple metadata types.
- Highly multi-threaded, cross-site data movement.
- Tolerant of high-latency for site-to-site connectivity.
- Supports up to 16 sites per share.
- File-granular data management (tiering, archiving, cloning, mobility…).
- Non-disruptive data mobility, even on live data.
- Support for extremely high-performance AI and HPC use cases on commodity storage, with support for Parallel NFSv4.2 with FlexFiles.

2.2. Key Components

Component	Description
Sites	Sites can be anything from geographically separated locations, all the way to fault domains within the same facility. Each "site" will host Anvils and can host some combination of DSX, Linux Storage Server (LSS) and 3rd-party NAS. Each site will also have its own set of objectives that dictate data placement for that site.
Anvils or Metadata Servers	Each site has metadata servers known as Anvils that are responsible for managing file system metadata (directory structure, file names, timestamps, etc.). These are typically deployed as HA pairs for site-level resiliency.
DSX Nodes	These servers have three roles: store, mover, and portal. The one we are mainly concerned with in the GFS context is mover, which handles the actual data transfer. Movers interact with cloud buckets and local storage to place file instances.
Shared Object Buckets	A central object storage location used for transferring file data between sites. This must be reachable by all sites participating in a given share.
Objectives	Policies defined by administrators to control data placement (e.g., "keep a copy in cloud," "keep a copy on SITE B").
Clients	User machines accessing the GFS via protocols like NFS, SMB, S3, and CSI.

© 2026 - Hammerspace

2.3. Recommended Use Cases

Distributed Workflows: Different sites work on the *same dataset* but at *different, non-overlapping times*. It is crucial to ensure that Replication is fully caught up before the next site begins active writes.

Burst-to-Compute: Compute and data are often not co-local. Hammerspace will orchestrate data to compute resources regardless of location whether it is a remote data center and/or public cloud (cloud bursting / hybrid cloud). Output data should ideally be written to *new, site-specific* locations or otherwise managed carefully to avoid conflicts.

Multi-site Collaboration: Enable follow-the-sun workflows where the primary "active" site performing read/write operations shifts geographically as the workday progresses globally.

Site-Specific Data: Workloads where each site operates on its own distinct set of files/directories, with minimal or no concurrent modification of shared datasets.

Data Tiering to the Cloud and/or Object Storage: Tier data seamlessly in the background without interrupting user access. This enables active data to remain on Tier 0 and Tier 1 while orchestrating data to Cloud and/or Object storage for dormant data as an active archive.

Multi-site Consolidation: Hammerspace global file system provides a single logical system across multiple sites allowing customers to consolidate their storage regardless of physical location.

2.4. Unsupported Use Cases and Configurations

Concurrent Updates to the Same File: Applications that attempt to modify the *exact same file* simultaneously from multiple active sites are unsupported. Examples include:

- Multiple batch schedulers writing to the same log file.
- Concurrent pip install into a shared directory can lead to writing to the same file(s)
- Shared home directories accessed concurrently from multiple VDI instances or workstations (e.g., Chrome profiles).
- Version control systems (e.g., Git) used concurrently on the same repository from different sites without explicit workflow controls.
- Engineering design applications (e.g., Windchill) or media editing software that expects synchronous, byte-range locking across sites for shared files.
- Application logging directories that are not unique to each site
- Large, read-write data files associated with databases or virtual machine images being concurrently run at multiple sites or changing sites with no time to allow data to move to the new site.

Considerations for the Following Configurations: The following setups can cause significant overhead and performance issues during snapshots due to the lack of offloaded cloning support.

- A system with a single NAS instance and no offloaded clones is not recommended for Replication.
- Systems not listed on the Hardware and Software Compatibility List require specific testing of their

 © 2026 - Hammerspace

offloaded clone capabilities before they can be considered fully supported.

IMPORTANT: Hammerspace does not provide synchronous, cross-site locking. Any application or workflow that relies on this will encounter issues.

2.5. Data Orchestration Replication Flow Example - Staged Workflow

1. **File Modification (SITE A)**: A user modifies a file on SITE A.
2. **Metadata Update**: SITE A updates the file's metadata (e.g., modify time, last use time).
3. **Metadata Checkpoints**: Regular checkpoints capture these metadata changes.
4. **Metadata Replication**: Metadata changes are replicated to other sites (SITE B, SITE C, etc.). This involves sending diffs computed from metadata checkpoints.
5. **Proactive Upload**: An objective on the owning site (SITE A) triggers an upload of the new dataset to the shared object volume(s).
6. **Data Download**: An objective on a remote site (SITE B) identifies that it should have the latest copy of the file. It then downloads a copy of the file from the shared object volume.
7. **Local File Copy**: The downloaded data from the shared object volume is used to create a local NAS copy of the file on SITE B.

 © 2026 - Hammerspace

Chapter 3. Prerequisites for GFS Deployment

Hammerspace can utilize block storage, existing Network Attached Storage (NAS) and Cloud/Object storage. It is installed as a single software load, including all components on bare-metal, virtual, and cloud-based environments.

3.1. Hardware Resources

Ensure each site has at least one installed Anvil metadata node (HA is strongly recommended for resilience) and at least one installed DSX data services node (two or more per site is recommended for resilience).

- Confirm that the underlying block storage of the Anvils has sufficient IOPS and low latency to handle metadata operations. Sub-optimal storage can severely impact Replication performance.
- Anvils must have sufficient CPU and memory to process metadata changes and manage metadata checkpoints, especially in environments with high file churn or many sites. Please work with your Hammerspace Sales Engineer to ensure you have adequate resources.

3.2. Network Connectivity

- Reliable and high-bandwidth network connectivity between all participating sites.
  - High-Bandwidth is defined as enough to transfer any new data and estimated changed data to or from any other site participants.
- Network connectivity between the Anvil nodes using four ports: **443, 9097, 9298, 9299**
- Proper firewall rules to allow communication between Anvils and DSXs across sites, and to the shared cloud bucket.

3.3. Shared Bucket

A pre-configured shared object-storage bucket accessible by all sites for data transfer. A central, non-site-specific bucket is preferred for Replication. This helps manage bandwidth and data transfers, especially when data is not needed at all remote sites. *Be aware of potential egress charges when using a bucket in a public cloud.*

Hammerspace object storage volumes (OSV) can support the following features:

- File chunking: 4MB chunk size, on by default. Can be disabled by adding the OSV in native mode, but use cases for disabling are not common.
- Deduplication: Hammerspace will deduplicate by matching SHA256 sums of chunks. This can not be disabled as there would be no advantage to disabling it.
- Compression: Compression can be enabled on OSVs. With some data sets this can save capacity and bandwidth, but may cost CPU cycles on DSX nodes.
- Encryption: If a Key Management Service (KMS) is configured, new objects placed in an OSV will be encrypted unless the OSV is configured with the `--no-encryption` option in the CLI.

 © 2026 - Hammerspace

The four features above are critical because shared OSVs should be added to each participating Hammerspace cluster with the same parameters. In particular, if the KMS is not the same for all clusters, clusters with different or no KMS will not be able to decrypt objects from sites with encryption enabled.

3.4. Global NFS Shares

There are no special requirements for NFS to work globally.

3.5. Global SMB Shares

For Windows and SMB data access, it is required that both sites are joined to the same domain/domain forest for the proper translation of Security Identifiers (SIDs) on both sites.

3.6. Global S3 Servers/Buckets

Each site can present data over S3 without any coordination required across sites for S3.

 © 2026 - Hammerspace

Chapter 4. Set Up Replication

Setting up Replication with **Hammerspace** involves configuring multiple sites, unifying them into a **Global File System**, and then establishing the **Data Orchestration Objectives** that govern how files move between them.

4.1. Step 1: Prepare the Infrastructure

Before configuring Replication, both sites must be ready and have access to the shared data plane.

1. **Deploy and Configure Sites:** Deploy the Hammerspace software (Anvils and DSX) at all required geographic locations (SITE A, SITE B, etc.). Ensure each site is fully operational. Reference the Hammerspace Installation and Licensing guide for guidance.
2. **Configure Local Storage:** Each site requires local file storage, such as a DSX with local block storage or a local NAS export to store data being used on that site. See the Hammerspace Configuration Guide for additional details.

4.2. Step 2: Configure a Shared Bucket

A shared object storage system will need to be added to each Hammerspace cluster to act as the shared repository for replicating file data content. Any supported object storage vendor/cloud (AWS S3, GCP GCS, or an on-premises S3-compatible system like MinIO) can be used as the shared bucket. If object storage is not available, then Hammerspace S3 functionality can also be used as the shared bucket.

The shared bucket is used for cross-site data transfers, but not for metadata traffic. A bucket must be added as a shared bucket from all participating sites. When a site is adding the shared bucket, a registration key is added to the bucket to simplify the setup of the Global File System for additional sites. The bucket can also be used for tiering and archiving as well as a general "overflow" bucket in case the local NAS runs out of space.

Only a single bucket is required (although using many is supported) and access to this bucket must be allowed for all Anvil and DSX nodes from all participating sites - access to the bucket can be limited to only the Anvil and DSX nodes as a security best practice.

Hammerspace also supports having multiple shared object stores on each site, which can be beneficial in reducing cross-site data traffic with some workflows. For resilience, having more than one shared object storage volume is recommended.

Note: If you need to configure advanced OSV features such as encryption, compression, and chunking (native mode), only compression is available in the GUI. Encryption and chunking must be configured through CLI or API. Remember to configure these features the same on all sites for the same bucket.

4.2.1. Add a Shared Bucket Using the GUI

1. Navigate to **Infrastructure** → **Storage Systems** and click on **+ Volume** for the desired object/cloud storage system. This will bring up the **Add Volume wizard**.

 © 2026 - Hammerspace

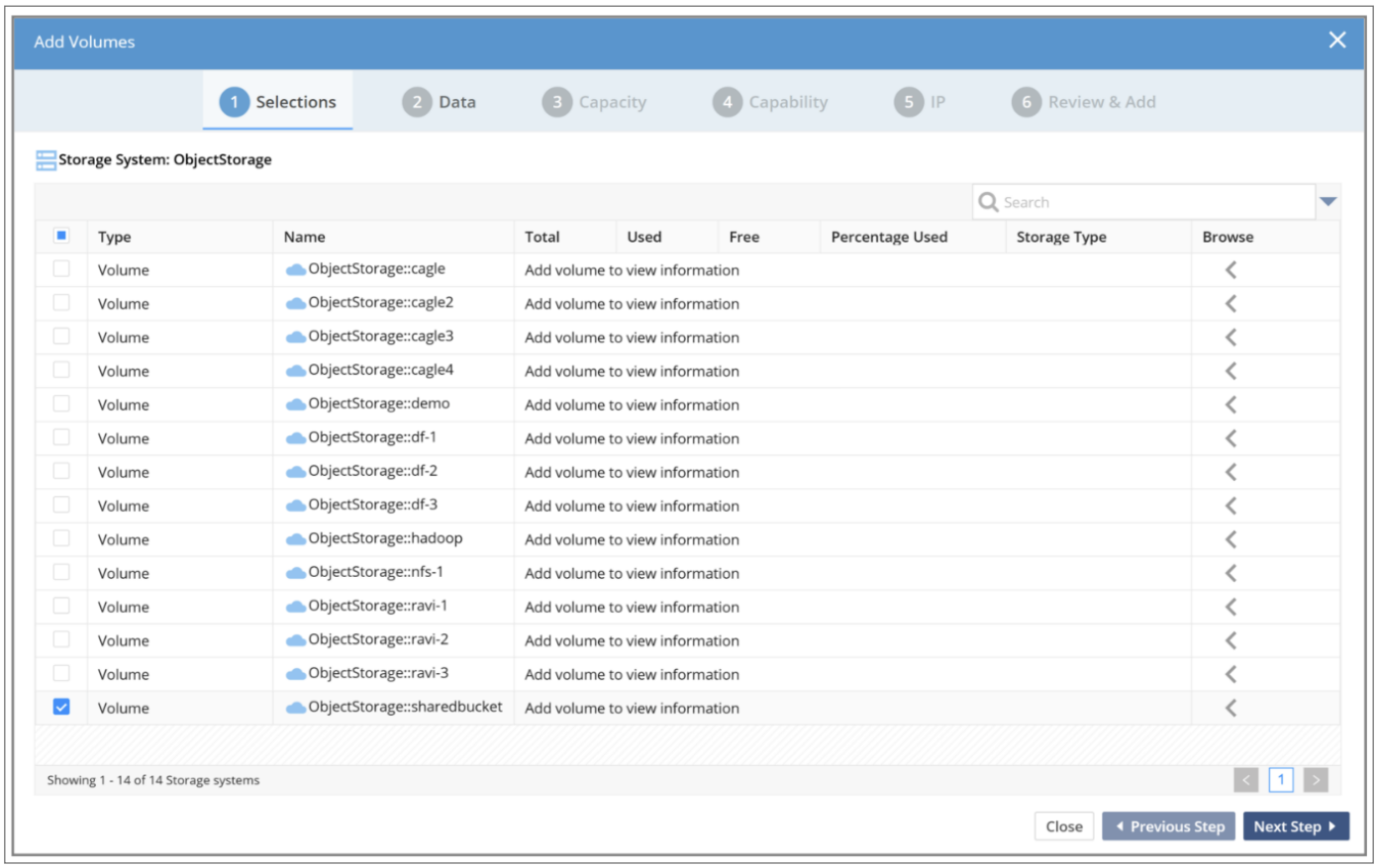

Figure 1. Add a shared bucket using the GUI

2. For the second, and any additional sites, the GUI will display a red warning icon, indicating that the site is in use by another Anvil. This is expected in a multi-site configuration. To proceed, make sure the Shared Volume checkbox is checked and click **Next Step**.

 © 2026 - Hammerspace

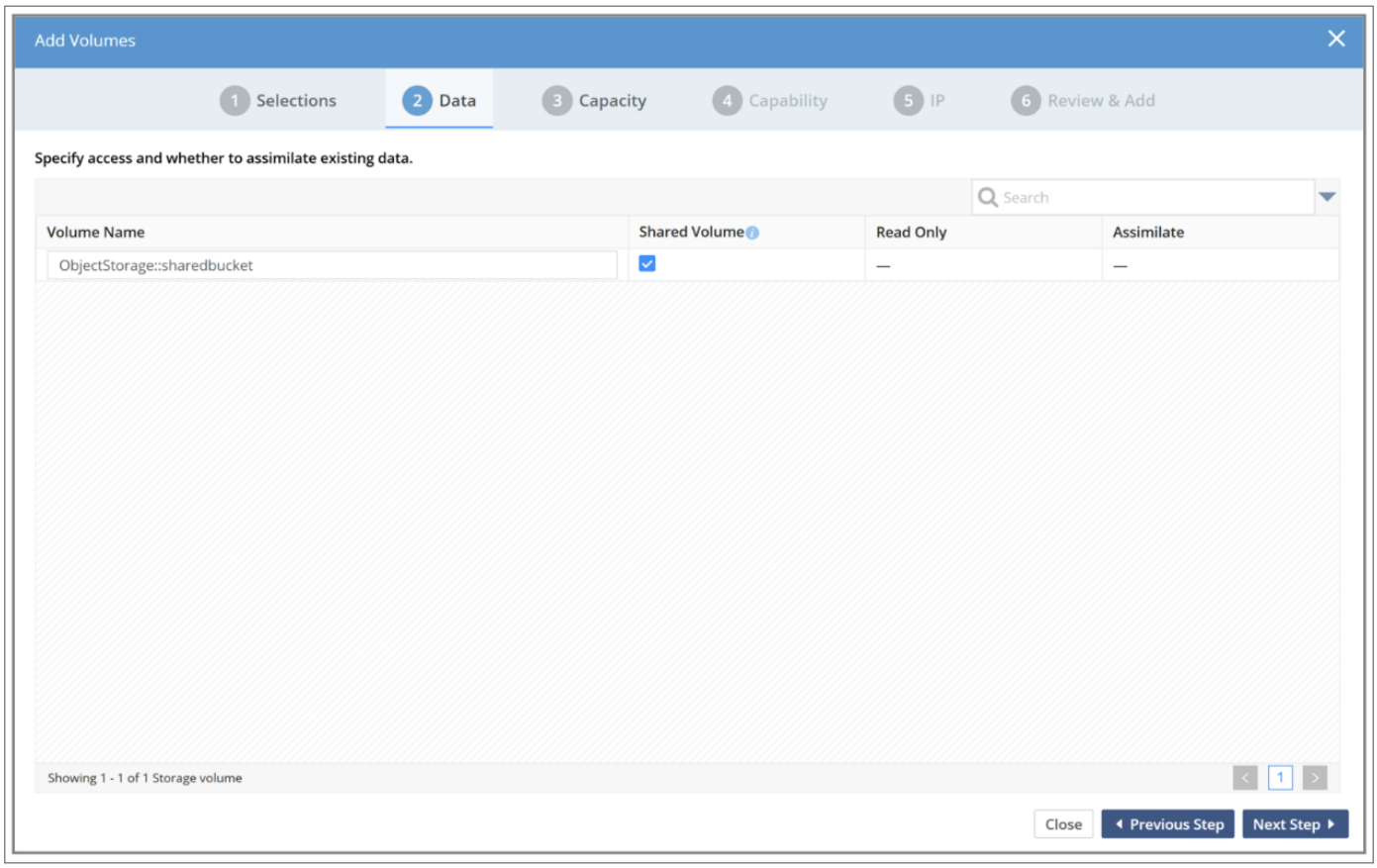

Figure 2. Add a shared bucket using the GUI

3. Complete the remaining steps and click **Add Volume**.

 © 2026 - Hammerspace

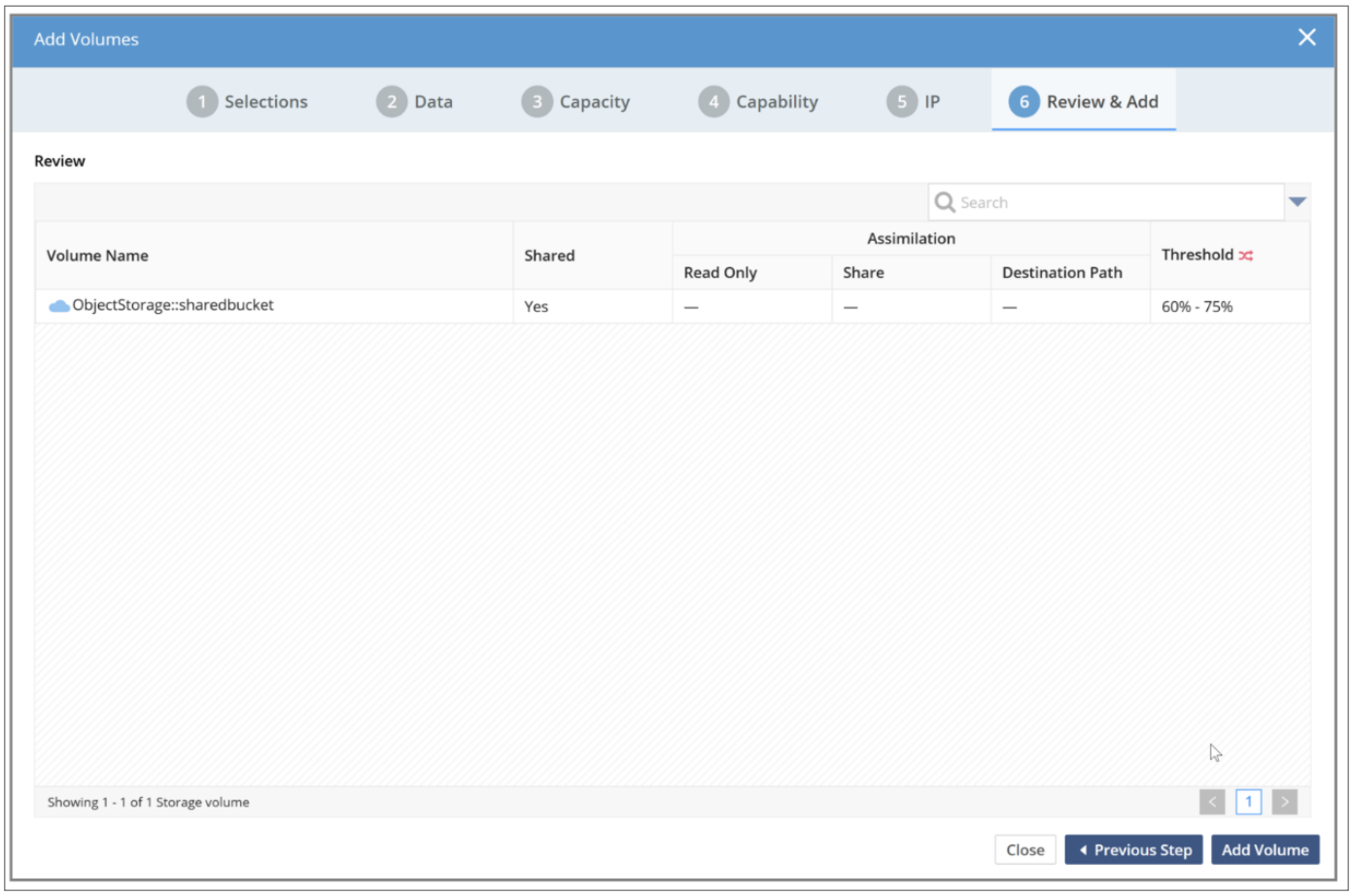

Figure 3. Add a shared bucket using the GUI

4.2.2. Add a Shared Bucket Using the CLI

You can add the `--shared` option to `object-volume-add` to indicate that this bucket is a shared resource. If this option is not used when first adding the bucket, then the second and third site will not be able to add the bucket as an object-volume. The option can only be configured when adding an object volume.

To keep the configuration simple, the same Storage System name has been used **(ObjectStorage)** on all three sites, and the bucket name is **sharedbucket**. This is not a requirement; each site can have its own names for Storage Systems and Volumes.

Configure the Shared Object Volume using the Admin CLI

Command:

```
object-volume-add --shared --node-name ObjectStorage --logical-volume-name sharedbucket
```

Expected output:

```
ID: 34f9b63a-1e07-47df-a245-1e5d7c152385
Name: ObjectStorage::sharedbucket
Internal ID: 268435473
```

 © 2026 - Hammerspace

```
Logical volume name: sharedbucket
State: OK
<...additional output truncated...>
```

Command:

```
object-volume-add --shared --node-name ObjectStorage --logical-volume-name sharedbucket
```

4.3. Step 3: Create a Global Share and Enable Replication

Create a new share or use an existing share and turn on share-replication, which will bi-directionally replicate all metadata changes from the share to all participating sites.

The initial GFS (metadata replication) setup is done on the site with the source share. Note that the same share name and path being configured for replication cannot exist on the target site. As part of the GFS create flow, the share will be created on the remote SITE and then be immediately populated with a near continuous stream of metadata.

The next step takes an existing share, named "Home," and creates a Global File System between SITE A, SITE B, and SITE C.

4.3.1. Create a Global Share Using the GUI

In the following example scenario, we will create a global share from the **Home** share using the Management GUI.

1. Navigate to **Data** and click on the **Shares** tab.

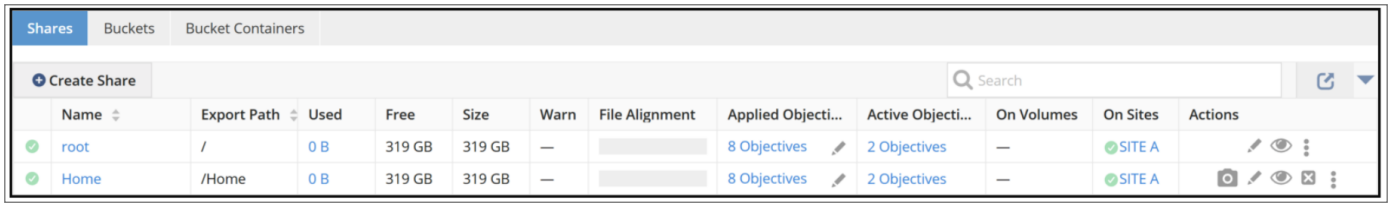

Figure 4. Create a global share using the GUI

2. In the row for the **Home** share, click the **Edit** (pen) icon under the Actions column to bring up the share-configuration dialog.

 © 2026 - Hammerspace

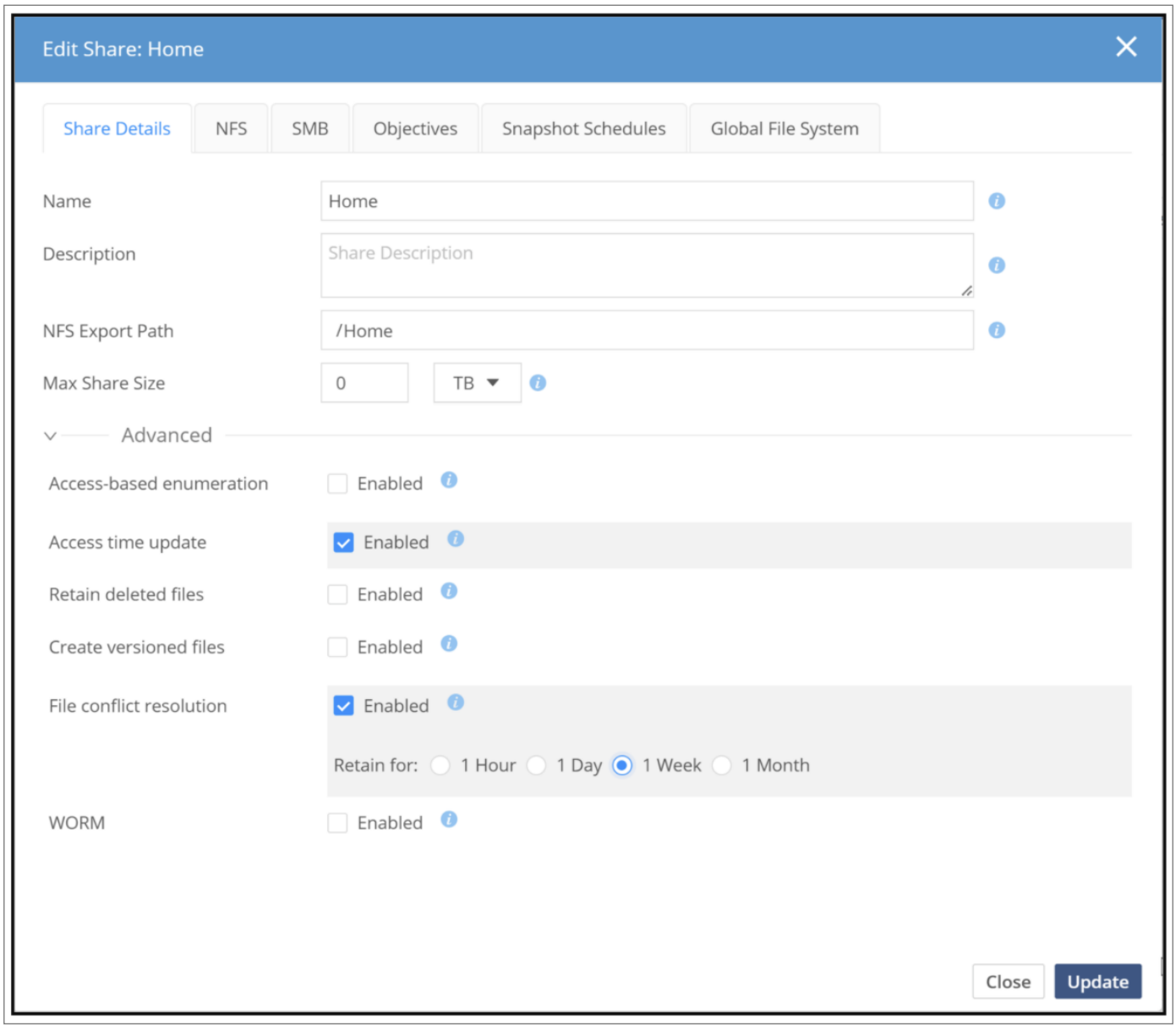

Figure 5. Create a global share using the GUI

It is highly recommended to use the File Conflict Resolution objective (log-xfer-1-<time>) to ensure that any files that change site ownership are stored as a version for the selected time period. The example below has checked the option to save files for a week.

3. Click the **Global File System** tab and **Add Participant**. Note that the pull-down menu will be pre-populated with any other sites that are registered to use the shared bucket. In this example, SITE B and SITE C.

 © 2026 - Hammerspace

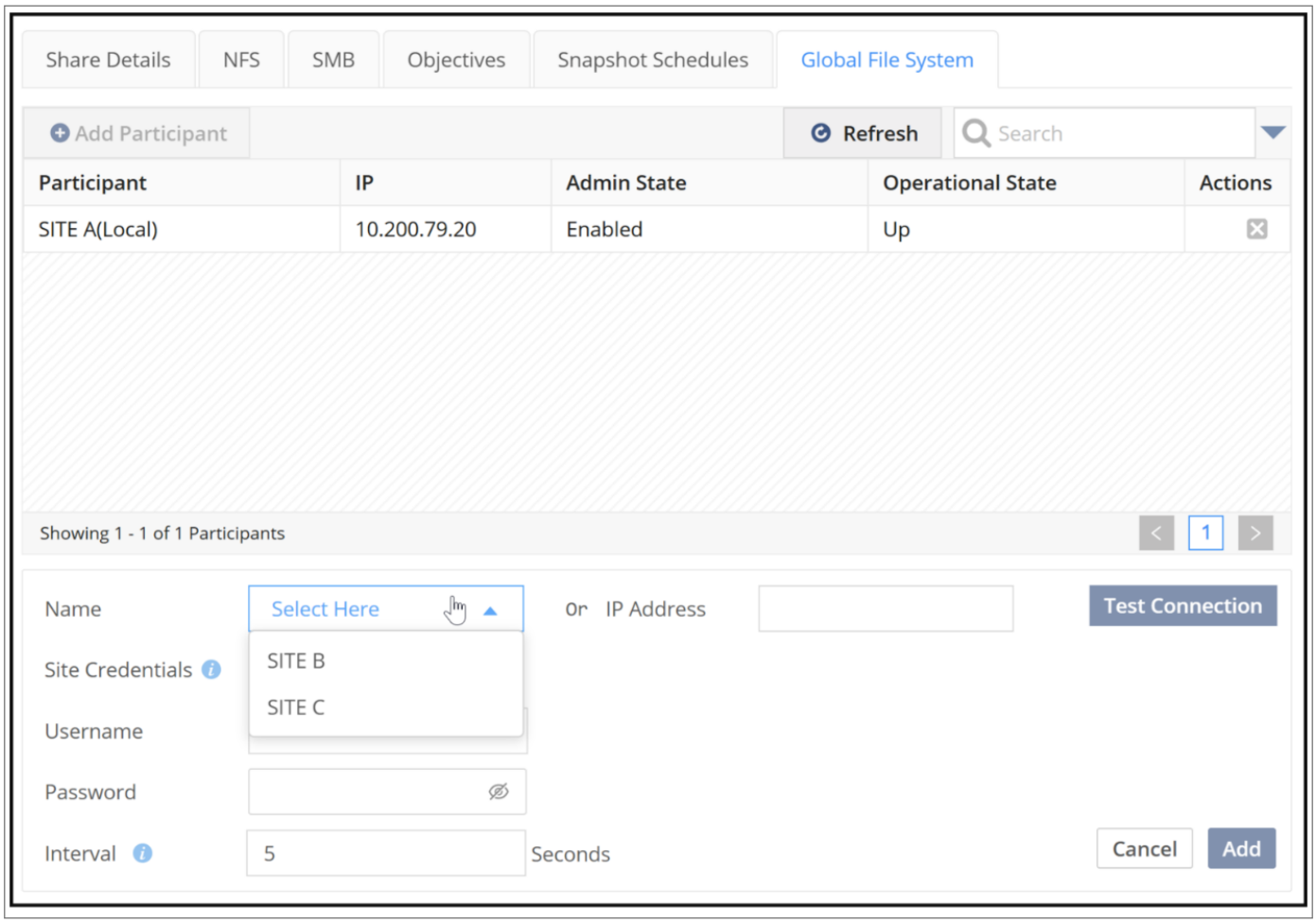

Figure 6. Create a global share using the GUI

4. Add SITE B and SITE C to our Global Home share by executing the following for SITE B and then again for SITE C:

- Select SITE B from the pulldown and the IP Address will be pre-populated. Check that connectivity works by clicking Test Connection.
- Fill in the credentials for admin-level user for SITE B. These credentials will be used to set up a trusted connection between the two sites with regards to management operations, specific to the Home share.

 © 2026 - Hammerspace

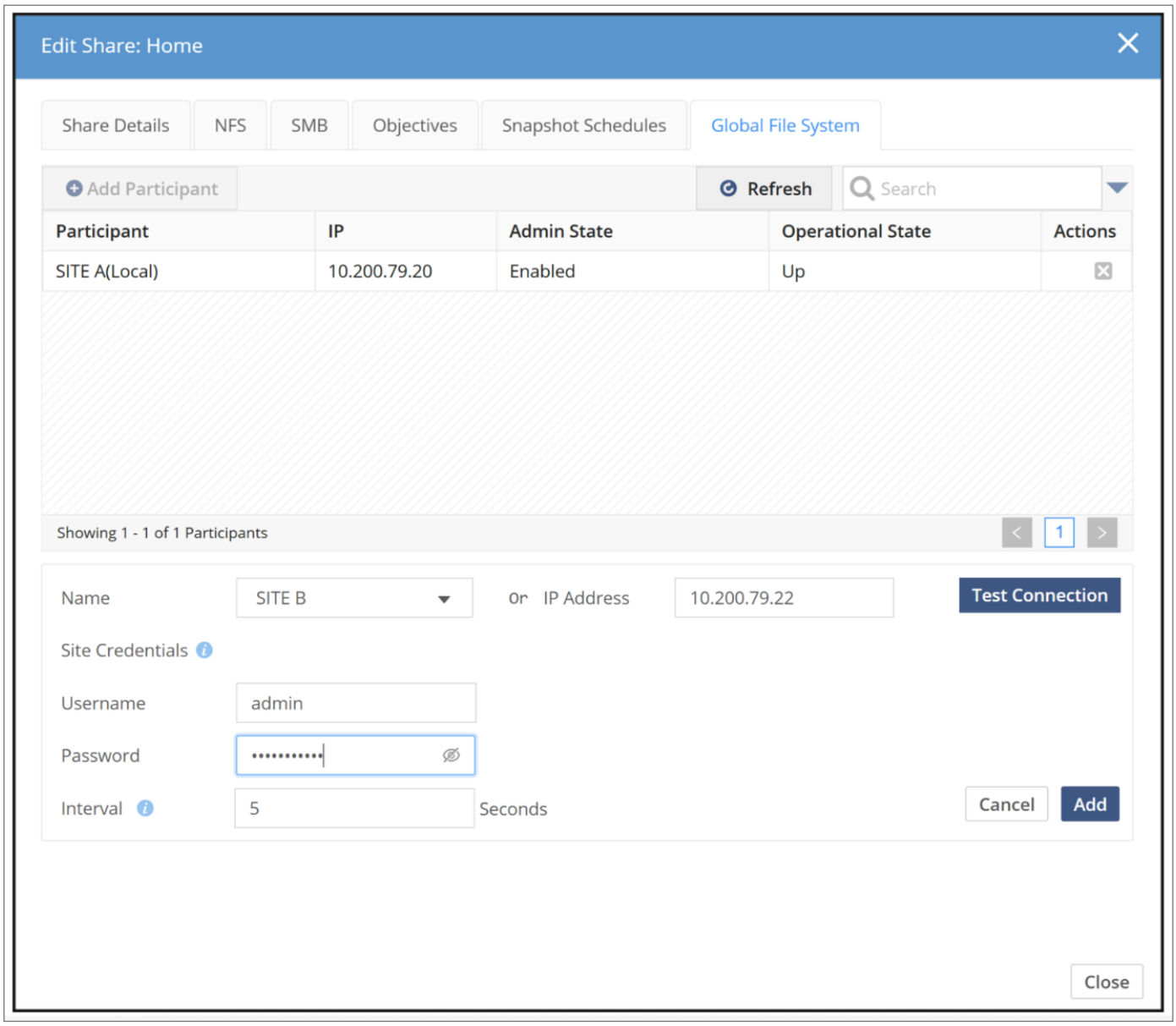

Figure 7. Create a global share using the GUI

5. Once complete, click **Add** to add SITE B. The add operation will create the share on the remote SITE And start metadata replication using point-to-point connection. This typically takes 15-30 seconds on most systems.

Once the Add operation has completed, the screen will display an additional site in the participant table.

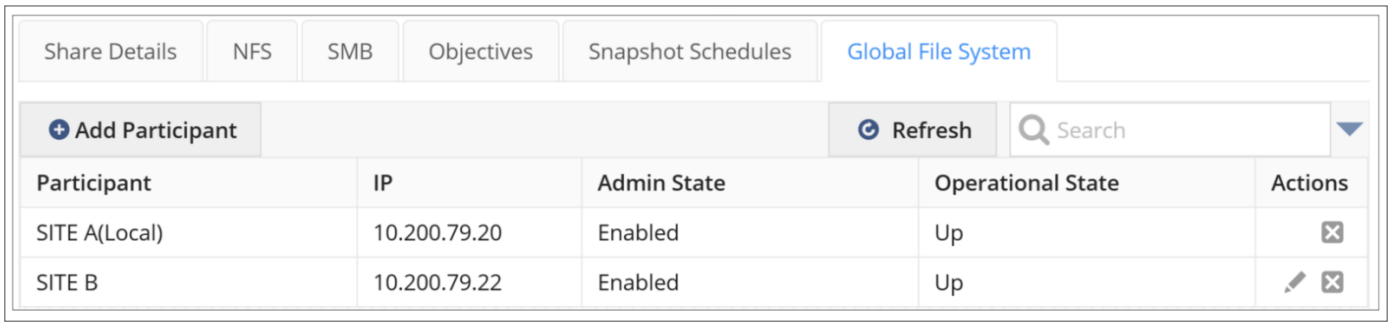

Figure 8. Create a global share using the GUI

© 2026 - Hammerspace

6. Repeat step 4 to add SITE C and have three participating sites for the **Home** share.

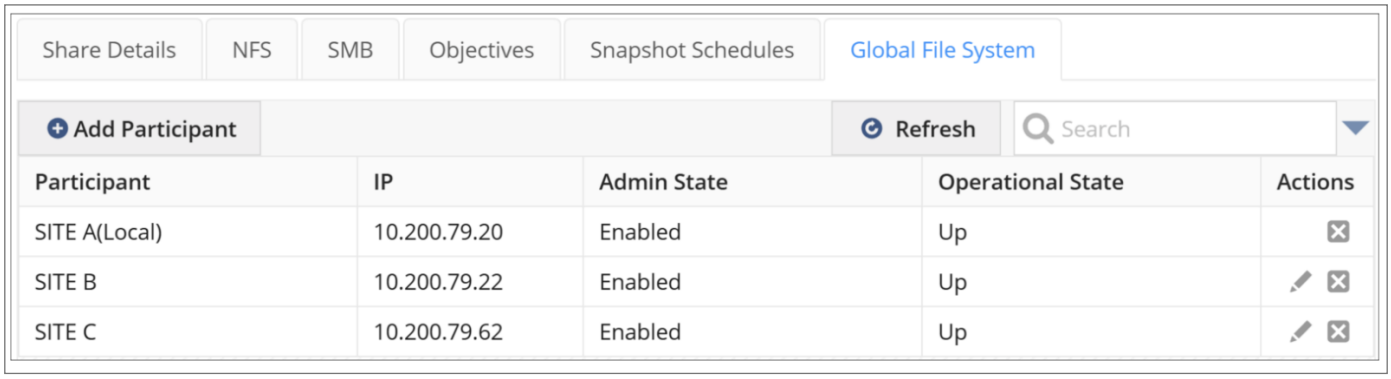

Figure 9. Create a global share using the GUI

The global share has now been configured and can instantly be used on the new sites.

7. Once the global share has been configured, verify that it is replicating metadata between all participating sites by navigating to the **Shares** tab. Click on the **Details icon** under the **Actions column**.

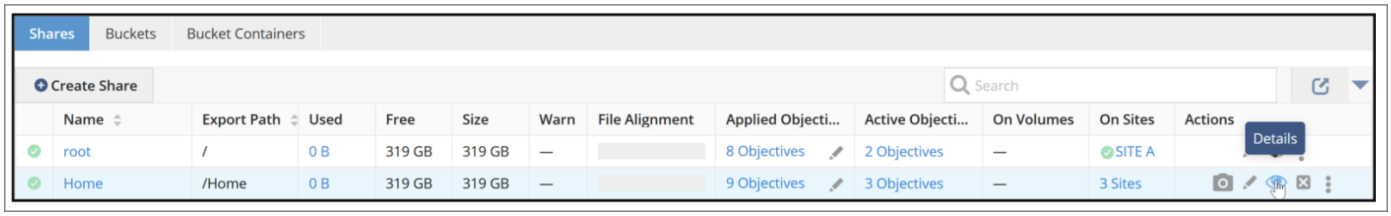

Figure 10. Create a global share using the GUI

8. Click on the **Global File System** tab to view Send and Receive latency.

NOTE: It is normal for these numbers to be 1.5x to 3x higher on a faster network, and higher when sites are located on very high latency networks. If the number is very large on an existing share, then it is likely that the initial metadata sync is taking some time. Either way, the shares can be used normally while the replication is being set up

.

© 2026 - Hammerspace

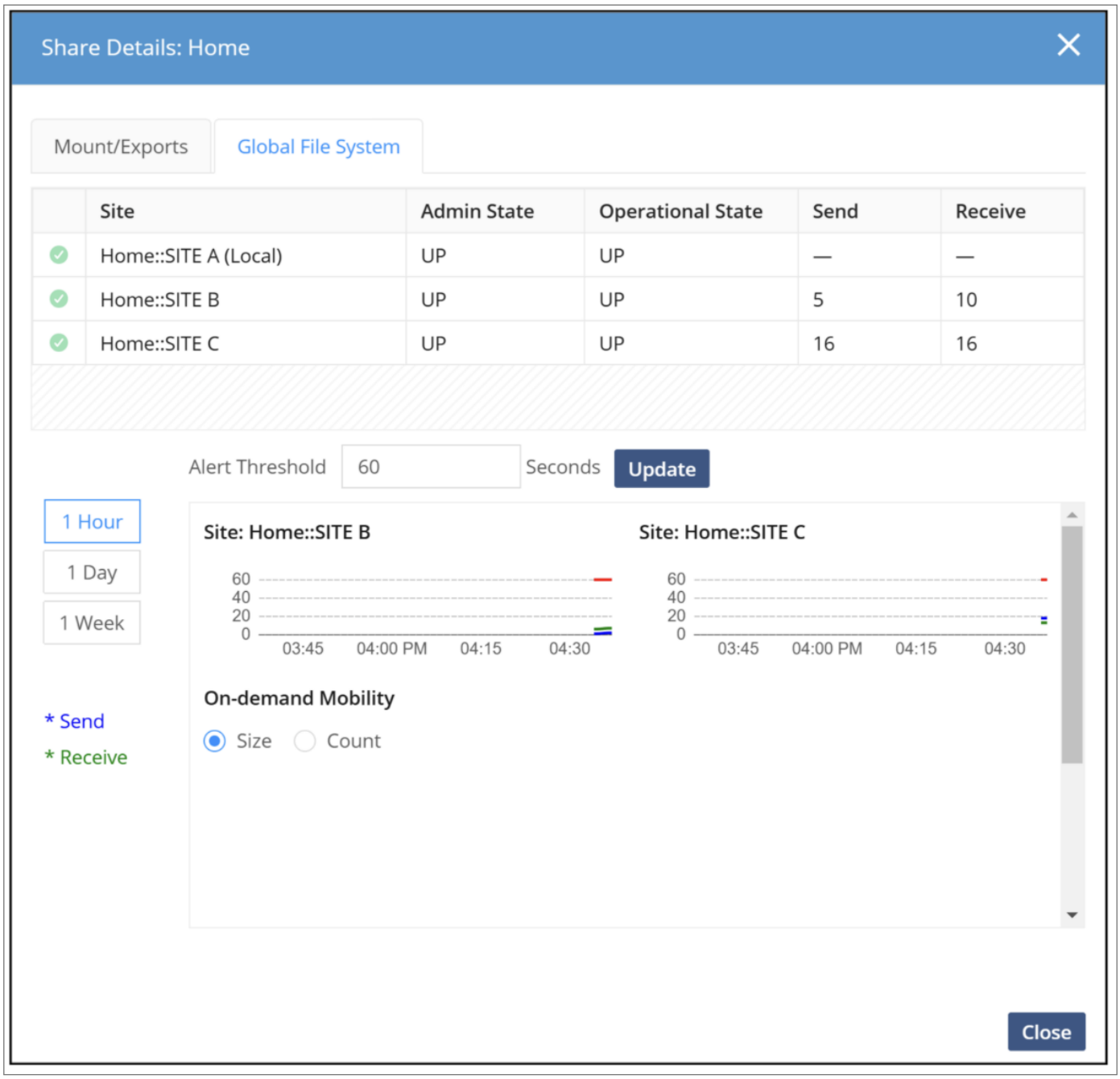

Figure 11. Create a global share using the GUI

9. Confirm that the directory structure and metadata are visible and consistent across all participating sites. This can be achieved by mounting the export or share at one of the site participants and viewing that files and directories appear as they do on the source.

- The default metadata Replication interval is **5 seconds**. This is the frequency at which sites check for and process metadata changes to send to their partners. This value can be adjusted, but should be chosen carefully based on network conditions and workload.

NOTE: Metadata sync is fast and continuous. The actual file data will only move once objectives are applied (outlined in the next section) or on-demand.

 © 2026 - Hammerspace

4.3.2. Create a Global Share Using the CLI

Run the following commands on SITE A to create the share and add the participants, SITE B and SITE C.

Command:

```
share-create --name Home --path /home
```

Expected output:

```
ID: a959d74f-0dcd-4cb2-b674-0d323ca74591
Name: Home
Internal ID: 1
Path: /Home
Lifecycle: CREATED
State: PUBLISHED
Size limit state: NORMAL
<...additional output truncated...>
```

Command:

```
gfs-participant-add --share-name Home --site-name "SITE B" --site-username admin --site-password
*********
```

Expected output:

```
Name: Home::SITE B
ID: 968b5b51-c521-495e-937a-36b0ffc6f9d4
Local share name: Home
Admin state: Up
Oper state: Up
Participant share internal ID: 3
Participant site name: SITE B
Participant site management address: 10.200.79.22
Participant site data address: 10.200.79.22
Participant ID: 1
Replication interval: 5 seconds
<...additional output truncated...>
```

Command:

```
gfs-participant-add --share-name Home --site-name "SITE C" --site-username admin --site-password
*********
```

Expected output:

 © 2026 - Hammerspace

```
Name: Home::SITE C
ID: 6fbdf5dc-93c5-4bfb-95b8-2ccf809f4921
Local share name: Home
Admin state: Up
Oper state: Up
Participant share internal ID: 3
Participant site name: SITE C
Participant site management address: 10.200.79.62
Participant site data address: 10.200.79.62
Participant ID: 2
Replication interval: 5 seconds
<...additional output truncated...>
```

4.4. Step 4: Define the Data Path with Objectives

With the Global File System, data is globally accessible by way of multiple standard protocols (NFS, SMB, and S3). The scope of data visibility is **share-granular** while data mobility is file granular. Because mobility across multiple sites leverages global deduplication (and/or compression and/or encryption), the data being transferred between sites may be less than the actual file content.

Each site is responsible for their own objectives to fully control data placement and performance without impacting other sites. This enables very dynamic usage of storage and supports a broad range of application workloads. Objectives dictate **what** data moves, **when** it moves, and **where** it is stored. These are the most critical settings for controlling Replication efficiency and data consistency.

4.4.1. A. Core Replication Objective

Apply a **Place On** objective to the replicated data set that directs files to the shared object storage. This objective initiates the creation of a copy on the shared object volume, triggering the upload process, effectively enabling Replication.

Objective	Target	Why it's Necessary
Place On	**Shared Object Storage Volume**	This mandates that the data content is uploaded to the central bucket.

4.4.2. B. Local instance/Performance Objectives

Define policies for local NAS tiers to manage performance, cost, and capacity at each site.

 © 2026 - Hammerspace

Objective	Target	Rationale
Keep Online	**Local NAS Volume (e.g., Tier 0)**	Ensures a copy of the file data is kept on the fastest local storage at a site where the data is actively being used. This maximizes local performance and avoids read latency.
Optimize for Capacity	**Local NAS Volume**	Do not maintain instances on local storage when no objectives require a local copy. The file data will be in the cloud, on object storage or on other sites. **Use with caution**, as it interferes with durability (see Step 4).

4.5. Step 5: Implement Durability and Consistency Controls

To mitigate the risk of data loss or corruption during Replication, especially for heavily modified data, apply the following controls.

Control Point	Action/Objective	Best Practice Rationale
File-Granular Protection	Ensure Versioning and Undelete objectives are set for critical shares (e.g., LogXfer = 1 Week).	This creates file-granular copies when data is overwritten or deleted, acting as a form of non-snapshot-based data protection against accidental changes.
Snapshot Scheduling	Schedule periodic Snapshots (e.g., daily) on the *active* primary site.	Provides a consistent point-in-time recovery reference. Note, in cases in which files change frequently, file-granular protection by way of versioning and/or undelete are recommended over higher-frequency snapshots

© 2026 - Hammerspace

Chapter 5. Key Configuration Parameters

5.1. Manage Data with Objectives

Hammerspace uses objective-based policies to automate data placement and other data services across all storage volumes under management. Unlike reactive policies, objectives enable customers to create plain-language business rules for how different classes of data should be managed. These objectives are *applied on each site* to specific directories or an entire share and act as the primary controller for data movement, including Replication.

Data can be dynamically moved on a file-granular basis between storage systems or sites to accommodate different performance requirements or other use cases, without interruption to users/applications. In addition, multiple instances of files can be replicated between sites for data protection, tiering, workflow staging, migration to new storage, or other needs in the background, without interruption to user access. These are not file copies, but instantiations of the same files, made possible by the shared Global File System metadata, which is kept synchronized and consistent across all sites.

In other words, since the Global File System spans all sites, users see the same data sets globally—across silos, sites, and clouds—regardless of where the physical bits of the files are stored today or may be moved to tomorrow.

No client software is required on user systems or application servers. They simply see their data as files or objects via the protocol they prefer, except now they have global visibility across all silos, sites, and cloud instances.

5.1.1. "LogXfer" Objective

By default, logxfer is an objective that’s applied to all shares and kicks in when a share becomes replicated. This feature creates a local snapshot of a file when ownership changes, acting as a form of data protection and a way to manually recover from user errors or conflicts.

5.1.2. "Keep on cloud" Objective

Configures the owning site to upload data to the shared cloud bucket. This is essential for enabling other sites to pull data and for disaster recovery scenarios.

5.1.3. "Keep on site X" Objective

Configures a specific site to maintain a local copy of data. This can be used to proactively pull data to a site, ensuring local availability even if the owning site goes offline.

5.1.4. Asynchronous Reservation tag-Based Objective

This capability allows a user to apply a tag to a file, effectively "reserving" it for exclusive write access by a specific site. A tag is set on the file, indicating the owning site. An objective is configured to make the file read-only on all *other* sites.

 © 2026 - Hammerspace

Note that this is an **advisory lock**; any user with write permissions can remove the tag. It does not provide hard, synchronous locking.

- **Replication Dependency**: This mechanism is **entirely dependent on Replication being in sync**. The tag must replicate to all other sites for the read-only objective to take effect.
- **Usage**:
  - **Set Lock**: Use an hs command to apply a tag (e.g., siteA_lock) to the file.
  - **Wait for Replication**: The user or an automated script **must wait** for this tag to replicate to all other sites before proceeding with modifications.
  - **Release Lock**: Use another hs command to remove the tag when done.

NOTE: Reference Appendix A to see example Replication use cases with objectives applied. For more information on setting up Objectives, see the {kb-objectives-url}[{kb-objectives-title}].

5.2. Snapshot Strategy

Snapshots can be user-initiated or scheduled,and are used primarily for data recovery from accidental deletes or overwrites, or for long-term archival, rather than for frequent versioning. The current best practice is to take a single snapshot once a day on one site. It is **not recommended** to take full snapshots with high frequency (e.g., hourly). Daily or weekly snapshots are generally sufficient.

- Taking a real snapshot initiates a copy-on-write process for files currently open for write, which can be resource-intensive, especially if underlying storage does not support offloaded cloning. This process is not instantaneous and introduces a "fuzz" in the "point-in-time" of the snapshot. Any files opened for write after that snap was created will also need to copy-on-write the data.

5.3. File Versioning and Undelete

While undelete and versioning are alternative options for data recovery, they are not currently recommended as a replacement for snapshots.

5.4. Data Durability & Availability

Replication configurations must prioritize data durability, especially for actively modified files. It’s recommended to use three-way client-side mirroring (CSM) for Tier 0. This ensures that even if one instance is dropped due to a COW Split (copy-on-write) or a node is in maintenance, there are still two active instances, providing data safety until re-silvering occurs. Note that 3-way CSM may also be beneficial for other storage tiers that do not possess RAID or Erasure Coding.

 © 2026 - Hammerspace

Chapter 6. GFS Best Practices and Guardrails

Overall Capacity Planning: It is important to consider the overall capacity required for:

- **Data required to remain online at each location**: Hammerspace will prefer NAS-based storage volumes, but will spill over to object storage tiers if capacity is needed. That is why it is essential to have spare capacity available if this occurs.
- **Any centralized, tiered data**: If the desire is to treat each site as more of a "cache", size each site's NAS volumes to accommodate the active working data, tiering to a centralized, shared object volume or bucket to optimize for capacity. Steps should be taken to ensure this object volume is highly durable and available.
- **All data that will be transferred between sites**: A shared object volume will be used for site-to-site data transfers. Site-to-site transfers will stall if there is not enough capacity within the share object volume to accommodate the data being requested.
  - It is recommended that enough capacity be configured to host 14-days worth of data transfer.
  - It is also recommended that a second bucket be configured with a higher *online-delay* objective. This allows for all transfers to prefer one bucket, using the second only if the first is unavailable, ensuring that data transfers are not interrupted due to a single bucket outage..

Data Deduplication: Deduplication can significantly reduce storage consumption, with savings of at least 10-20% and sometimes up to 50% depending on the data set. *Dedupe is on by default and it's recommended to keep it on*.

Data Compression: While enabling compression can reduce data significantly, it will also bring additional load to the DSX nodes performing said compression. Care should be given to sizing the DSX nodes appropriately to accommodate this additional load.

Coordinated Garbage Collection (CGC): CGC is a process that removes objects from buckets that no longer have references from any site, which helps to reduce bucket utilization and prevent object space from growing unchecked.

Consistency is Key: If using encryption, ensure that the passphrase or KMS key is consistent across all sites using the same bucket. Inconsistent encryption can prevent sites from decrypting files and will cause data orchestration to fail.

6.1. Addressing Specific Protocol Challenges

SMB workflows, particularly those involving file creation and renaming (e.g., "New Word Document" → rename), generate multiple metadata operations in quick succession. When combined with Replication delays, this can easily lead to race conditions and metadata conflicts (e.g., conflict files, lost and found entries). ***Implement very strict guardrails for SMB use in multi-SITE Active-active scenarios. It is generally best confined to active-passive or read-only access in replicated environments.***

6.2. Handling Replication Delays and Disconnections

Replication Falling Behind: This can be caused by under-resourced hardware (slow disks, insufficient

 © 2026 - Hammerspace

CPU/memory), network issues (latency, bandwidth), or workload patterns (high churn, concurrent writes).The goal is to keep Replication delay under an hour.

Extended Disconnection: While GFS can run indefinitely disconnected, reconnecting after a long period (especially if active changes occurred on multiple sites) significantly increases the risk of complex metadata conflicts.

For active-active scenarios, extended disconnections should be avoided. If a site is disconnected for a prolonged period, treat it as a potential source of data inconsistencies upon reconnection, and be prepared for manual reconciliation.

6.3. Metadata and Data Collisions

Metadata and data collisions occur when two or more sites modify the same file or directory simultaneously. Another possible scenario is during situations where sites are unable to reach one another, yet workers at each site continue to work on their local instances of the same data. In this case manual steps may be necessary to reconcile these updates.

Data Collisions: When a file is being actively written to on one site, it cannot be accessed or uploaded to another site. A remote site attempting to read or write to that file will hang until the file is closed by the primary writer. This can cause delays for the end user. Note that this should not be confused with snapshot activity in which any open files will still be protected in the snapshot. If files remain open and access is needed on other sites, these files can then be accessed from the snapshot.

Conflict Files: In cases of simultaneous writes to the same file, a conflict file is created. While this prevents data loss from a merge, it indicates a problematic workflow. Where possible, forwarding workloads to the same site or location can significantly reduce this phenomenon.

6.4. Handling Conflicts

When using data across multiple sites, for those very far apart from each other, there are several considerations that come into play that affect the end user experience:

- How fast will I see new files?
- What happens if I modify the same file on both sites at the same time?
- What if the sites are disconnected from each other for an extended period?

This often comes down to how ownership and conflict resolution is handled. With Hammerspace, there are several options that can be applied to ensure various workflows can be supported.

As previously discussed, objectives are used to help manage data across sites. Objectives can place data locally and ensure that local copies are kept up to date when changes happen to files/objects on other sites. Objectives can also determine the behavior for such files/objects when conflicts occur:

- **log-xfer-<time>** objective is used to ensure that a version of the file is retained before the file changes owner. File ownership is determined by the site that last wrote to it. File ownership is important if another site needs to use the data, but it does not have a local copy of the data, it will reach out to the site that owns

 © 2026 - Hammerspace

the data and orchestrate mobility to the local site.

- **undelete-<time>** objective can help save files for a specific period if they are deleted. Undelete enables a quick recovery when files are accidentally deleted.

The Global File System uses an eventually consistent model, which relies on conflict resolution logic for when there are conflicts. Cross-site file locking is not supported.

6.4.1. File Conflict Scenario #1: Creating the Same Filename Across Multiple Sites at the Same Time

If the same file is created or modified on multiple sites during the same replication interval (or when sites are disconnected from each other), then the Global File System will automatically create two or more versions of the file, a site-local file with the original name, and a second file representing the remote version of the file. If more than two sites collide at the same time, there will be versions from each site.

Example

In the example shown in the table below, we create a file with the same filename, but a different size file, from each site to help explain the behavior. The commands are run from a local Linux client located on each site, with the share mounted under `/global`.

SITE A (ID: 0)	SITE B (ID: 1)	SITE C (ID: 2)
`# cd /global` `# dd if=/dev/urandom of=file bs=1024k count=1 conv=notrunc`	`# cd /global` `# dd if=/dev/urandom of=file bs=1024k count=5 conv=notrunc`	`# cd /global` `# dd if=/dev/urandom of=file bs=1024k count=9 conv=notrunc`

Wait until the replication interval has passed and view the directory with `ls -l`. Note that the NFS client will cache directory contents which could temporarily display information that is out of date. Keep in mind that some less-relevant columns have been removed due to space limitations.

Note the highlighted file across shares:

 © 2026 - Hammerspace

SITE A (ID: 0)	SITE B (ID: 1)	SITE C (ID: 2)

Command:
`# ls -l`
Expected output:
`total 15360`
`1048576 Feb 28 00:44 file`
`5242880 Feb 28 00:44 file[#S=1]`
`9437184 Feb 28 00:44 file[#S=2]`
	Command:
	`# ls -l`
	Expected output:
	`total 15360`
	`5242880 Feb 28 00:44 file`
	`1048576 Feb 28 00:44 file[#S=0]`
	`9437184 Feb 28 00:44 file[#S=2]`
		Command:
		`# ls -l`
		Expected output:
		`total 15360`
		`9437184 Feb 28 00:44 file`
		`1048576 Feb 28 00:44 file[#S=0]`
		`5242880 Feb 28 00:44 file[#S=1]`

As shown in the table, above, the same file name exists on all the sites however the conflict file version from the other sites is automatically renamed to FILENAME[#S=SITE ID]. This is done to ensure that the file being used locally does not suddenly change name and it also ensures that other sites do not overwrite that local file content.

To resolve a conflict, simply copy the renamed file to the original file name and delete the automatically named file if it is no longer needed.

6.4.2. File Conflict Scenario #2: Writing to an Existing File on Multiple Sites at the Same Time

While file metadata replicates to all sites, the actual file data (the file contents) is only replicated when requested, either explicitly through objectives or implicitly when the file is accessed. Objectives ensure that each site has a local copy of the data before writing to the file.

By default, the last writer will win when two sites write to the same file within the same replication interval. To help protect against user mistakes, the Global File System has the capability to preserve previous file content. This functionality must be turned on, which is done by using an objective.

Since the site-ownership ends up creating additional copies of files (very similar to Undelete and Versioning functionality), these extra copies have an expiration date, which is the time set by the objective. If snapshots are taken before files are deleted, then the files will remain within the snapshots until either the snapshot retention period or the file expiration period ends, whichever is the longest.

Site Ownership Objectives

The name of the objective indicates how long these files are retained in the live tree. When a retained file is copied, the retention is not copied with the file.

- log-xfer-1-hour
- log-xfer-1-day

 © 2026 - Hammerspace

- log-xfer-1-week
- log-xfer-1-month

The retained files are available in .snapshot/current,and each filename is made unique by both a timestamp and the originating site identifier being appended to the end of the files.

Command:

```
# ls -lt .snapshot/current/
```

Expected output:

```
-rw-r--r-- 1 root root 217 Feb 28 07:33 file
-rw-r--r-- 1 root root 174 Feb 28 07:30 file[#M=20200916.212518.75035 #S=1-5]
```

In the example above, the file entry represents the live tree version, and the second entry represents the saved version of the file with the timestamp appended to the file. S=1-5 translates to Participant ID (site) 1 and 5 is the object ID in the file system.

The site ID can be determined from the mounted share itself if the {kb-hstk-url}[{kb-hstk-title}] is installed, or it can be determined via the admin login using the `share-list --name <share-name>` output.

6.5. User Education for Workflows

End users often lack awareness of the underlying eventually consistent nature of a GFS. For workflows that involve shared data and multiple sites, administrators must implement:

- **Workflow Integration:** For complex multi-site, multi-user scenarios (e.g., in the media & entertainment industry), integrate with third-party workflow management systems (e.g., ShotGrid via ShotHammer plugins) to enforce check-in/check-out, manage file access, and coordinate changes.
- **Site-Specific Workspaces**: Encourage users to work within designated site-specific directories (e.g., /home/user/siteA_projects) when performing active writes to avoid conflicts.
- **Clear Guidance**: Provide explicit instructions on how users should interact with shared data across sites or when to use site-specific copies.
- **Asynchronous Reservation (Advisory Locking)**: Implement advisory locking mechanisms to signal intent to modify a file. Users must understand that this is **advisory** and relies on Replication being in sync. It requires a manual process to acquire and release the "lock" and verify its status across sites.

 © 2026 - Hammerspace

Chapter 7. Monitoring and Alerting

Replication is eventually consistent; therefore, continuous monitoring is critical to ensure predictable performance and consistency.

- **Monitor Latency:** Actively monitor **Replication Lag** metrics (available via the API/CLI/GUI/Prometheus) to ensure the Metadata Path remains responsive and that lag does not consistently exceed key thresholds. (Note that by default, an alert will be triggered if replication lags beyond 60-seconds, or 5x the desired replication interval, whichever comes first.) The Hammerspace GUI provides a "Global File System" section where administrators can view Replication latency charts (1-hour, 1-day, 1-week views) for each share.
- **Alert Thresholds**: Configure alert thresholds within the GUI for Replication latency. If latency exceeds this threshold, an alert will be displayed in the UI.
- **Grafana:** Visually track Replication health in real-time with metrics from Prometheus and proactively detect delays and failures before they become critical with customized dashboards and alerts. For enabling the exporters and installing the dashboards, see *Monitoring Cluster Health* in the Hammerspace Administration Guide.
- **API Access**: All data displayed in the GUI is accessible via APIs, allowing for custom monitoring solutions.
- **SNMP/Syslog Integration**: Leverage SNMP or Syslog to integrate Hammerspace alerts into existing enterprise monitoring systems.

Reference Appendix B to see common errors that may be encountered.

© 2026 - Hammerspace

Chapter 8. Conclusion

The Hammerspace Global File System architecture provides a robust, resilient, and high-performing solution for managing geo-distributed data. The platform's ability to maintain a consistent GFS is built on four key pillars:

- A **resilient replication engine** that ensures data integrity at scale.
- A **multi-tiered conflict management framework** that provides both proactive and reactive tools for collaboration.
- A **robust lifecycle management system** that simplifies the non-disruptive creation and decommissioning of sites.
- An **intelligent data orchestration engine** that separates metadata from data to enable policy-driven data placement and management.

Hammerspace's architectural design makes it a perfect fit for demanding enterprise workloads that require a truly active-active global data platform.

© 2026 - Hammerspace

Appendices

© 2026 - Hammerspace

Appendix A: Additional Resources

- {kb-objectives-url}[{kb-objectives-title}]
- Hammerspace Configuration Guide
- Hammerspace Installation and Licensing Guide
- {kb-hstk-url}[{kb-hstk-title}]
- Grafana Dashboards for Hammerspace

© 2026 - Hammerspace

Appendix B: Appendix a - Replication Examples

B.1. Example #1

Customer has two sites - a primary site (SITE A) and a DR site (SITE B). The DR site is only used if there is an issue with the primary site.

Customer utilizes a Pure Flashblade for online file storage and a Pure S3 at each site, on the same Flashblade system, for an archival and space-efficient tier.

The following three objectives are applied to all shares. This example is for SITE A, but SITE B uses an identical strategy. Each cluster places all online files and file snapshots in the Pure Flashblade buckets at both sites, a Hammerspace best practice for customers who leverage GFS for DR. Live file data that has been used within the last 90 days is kept online on a Pure Flashblade NFS volume. Customer does not use Hammerspace DSXs for storage; they are only used as a data mover.

[Name: place-on-SITEAPureS3, Applicability: 0, Removable: true]

[Name: place-on-SITEBPureNFS, Applicability: IS_LIVE AND LAST_USE_AGE<90*DAYS, Removable: true]

[Name: place-on-SITEBPureS3, Applicability: TRUE, Removable: true]

To complement their data protection strategies, snapshots are enabled, as are versioning, undelete, and log-xfer objectives. The versioning and undelete objectives utilize exclusions that avoid targeting unnecessary files.

[Name: versioning-1-day, Applicability: SUM(\{||#A=FNMATCH((\{".tmp";"**.TMP";".docx";" .xlsx";"~**.pptx"})[ROW],NAME)}[5])==0, Removable: true]*

[Name: undelete-1-day, Applicability: SUM(\{||#A=FNMATCH((\{".tmp";"**.TMP";".docx";" .xlsx";"~**.pptx"})[ROW],NAME)}[5])==0, Removable: true]*

B.2. Example #2

Customer has two sites, a primary and a DR site. The DR site is not used unless the primary site is unavailable.

Customer uses Hammerspace DSXs for online file storage, and Google Cloud object storage for both a DR archive, and to down-tier data that has not recently been used. Once the indicated time elapses, the online instance is dropped, leaving only the instance in the Google Cloud storage.

Customer uses Hammerspace tiers to direct the placement of files onto their DSX servers.

The dsx_mirror tier is configured to create two instances, while the dsx_any objective creates one instance on any available DSX. The shares use both tiers, but under different conditions as described below.

Name: dsx_any

 © 2026 - Hammerspace

Place on:

[node: dc1dsx2.company.com; node: dc1dsx1.company.com; node: dc1dsx3.company.com]

Name: dsx_mirror

Place on:

[node: dc1dsx2.company.com; node: dc1dsx1.company.com; node: dc1dsx3.company.com]

The following three objectives are applied to all shares.

For the first 24 hours after a file is created or modified, two online instances are kept to ensure the file has time to be copied to Google Cloud storage:

Name: dsx_local_mir_24h

Expression: IF MODIFY_AGE<24*HOURS&&DATA_ORIGIN_LOCAL THEN \{SLO('dsx_mirror')}

Locally edited or created files are kept online for two weeks:

Name: dsx_local_2weeks

Expression: IF MODIFY_AGE<2*WEEKS&&DATA_ORIGIN_LOCAL THEN \{SLO('dsx_any')}

FIles created or edited at the DR SITE Are also kept online for 24 hours:

Name: dsx_remote_24h

Expression: IF MODIFY_AGE<24*HOURS&&DATA_ORIGIN_REMOTE THEN \{SLO('dsx_any')}

One of the following two objectives are used on specific shares to ensure that if a locally created or modified file has not yet been copied to Google Cloud (!HAS_INSTANCE) and (||) has been modified within less than 24 hours, an instance is kept on a DSX.

Name: sharea_dsx

Expression: IF DATA_ORIGIN_LOCAL&&(!HAS_INSTANCE_ON("google cloud nearline::finance_records")||MODIFY_AGE<24*HOURS) THEN \{SLO('dsx_any')}

Name: shareb_dsx

Expression: IF DATA_ORIGIN_LOCAL&&(!HAS_INSTANCE_ON("google cloud nearline::hr_records")||MODIFY_AGE<24*HOURS) THEN \{SLO('dsx_any')}

The following two objectives are used on specific shares to ensure that an instance of each file is placed on one specific Google Cloud storage container, while the other ensures that they will not be placed on a different storage container.

[Name: place-on-google cloud nearline::hr_records, Applicability: ALWAYS, Removable: true]

 © 2026 - Hammerspace

[Name: exclude-from-google cloud nearline::finance_records, Applicability: ALWAYS, Removable: true]

Note: If a cluster has available object storage locations, and the default optimize-for-capacity objective is present, it will be downtiered to any allowed object storage container if no objective compels a file to stay online. This happens after 5 minutes by default, which is why it is important to define objectives that keep files online for as long as is needed or wanted, especially when utilizing object storage.

© 2026 - Hammerspace

Appendix C: Appendix B - Common Errors

C.1. Gfs-participant-add: Fails to Add a Site

Example Error Message

Command:

```
gfs-participant-add --share-name Apps --site-name "SITE Q" --site-username admin --site-password
**********
```

Expected output:

```
gfs-participant-add: site 'SITE Q' not found.
```

This error indicates that the site cannot find the SITE Q identifier in the configuration that has been exchanged between the sites. This indicates that either the `--site-name` input was mis-spelled (note that input is case sensitive) or that a shared object storage volume has not been set up correctly between the sites.

C.2. Gfs-participant-add: SITE-B Is Already Participant 1 of Share Home

Example Error Message

Command:

```
gfs-participant-add --share-name Home --site-name "SITE B" --site-username admin --site-password
***********
```

Expected output:

```
gfs-participant-add: Site SITE B is already participant 1 of share Home.
```

The share Home is already replicating to the site labeled SITE B.

C.3. Error Message: Bad Credentials

Example Error Message

Command:

```
gfs-participant-add --share-name Apps --site-name "SITE C" --site-username admin --site-password *****
```

 © 2026 - Hammerspace

Expected output:

```
gfs-participant-add: Unable to ascertain the software version running at 10.200.79.62. Cause: Remote site
'10.200.79.62' returned error: bad credentials
```

The username or password was incorrect when running `gfs-participant-add`.

C.4. Share-replication-create: Remote Site Error: An Entity (Share) Named 'Home' Already Exists

Example Error Message

Command:

```
gfs-participant-add --share-name Apps --site-name "SITE C" --site-username admin --site-password
*********
```

Expected output:

```
Status message: Remote site '10.200.79.62' returned error: A share with name 'Apps' exists. Share names
are case-insensitive.
```

It is not possible to establish a replication relationship with an already-existing share on the remote site. It is recommended to rename and move the share on the remote site to address this.

© 2026 - Hammerspace