Great idea — dividing the work into **focused subcommittees** is essential to tackle a complex standard like **GDSN (Global Distributed Single Namespace)** efficiently and to ensure cross-vendor collaboration without getting bogged down by scope creep.Here’s a well-structured breakdown of suggested **subcommittees**, along with a description of their **responsibilities, deliverables**, and potential overlaps to coordinate with other groups:**:jigsaw: Proposed Subcommittees for GDSN Standardization1.**  
**Namespace & Pathing CommitteeFocus:** Defining the structure, syntax, and semantics of the GDSN namespace.

* Define canonical path format (e.g., /gdsn/org/region/user/path)
* Support for hierarchical, flat, and hybrid models
* Define naming collisions and resolution rules
* Internationalization (i18n), versioning, and path metadata
* Alias/shortcut support (e.g., symlinks or virtual mounts)

:repeat: Coordinates with: API, Backend**2.**  
**API & Access Protocols CommitteeFocus:** Standardize APIs and access protocols for interacting with GDSN.

* RESTful APIs for listing, resolving, and managing namespace entries
* FUSE/gRPC/NFSv4-compatible client interfaces
* Object and file semantics (e.g., directory listing, file ops)
* Batch operations, consistency hints, and latency controls
* Compatibility layer mappings (S3, POSIX, etc.)

:repeat: Coordinates with: Namespace, Backend, Security**3.**  
**Backend & Storage Integration CommitteeFocus:** How GDSN maps to real storage systems.

* Define plugin architecture or adapters for:
  + POSIX filesystems (GPFS, Lustre, XFS, etc.)
  + Object stores (S3, GCS, MinIO, etc.)
  + NAS/SAN systems
* Metadata caching and consistency models (e.g., eventual vs strong)
* Namespace registration mechanism (how backend mounts are registered)

:repeat: Coordinates with: Namespace, API, Security**4.**  
**Identity & Access Control (Security) CommitteeFocus:** Standardize security, authentication, and access control.

* Identity federation (SSO, IAM, AD, LDAP, etc.)
* Unified ACL model (POSIX + S3 + NFSv4 ACLs)
* Delegation and impersonation support (e.g., on-behalf-of)
* Token formats (e.g., JWT, OAuth2) and access audit logging
* Encryption support and trust model for distributed namespaces

:repeat: Coordinates with: API, Interactions, Backend**5.**  
**Namespace Interactions & Referrals CommitteeFocus:** Standardize how clients discover and traverse the global namespace.

* Namespace referral mechanisms (e.g., HTTP 3xx, NFS referrals)
* Discovery protocols (how clients find roots or nearby namespace services)
* Client failover, retries, and redirect logic
* Integration with DNS or custom registry services
* Cache invalidation policies

:repeat: Coordinates with: API, Namespace, Security**6.**  
**Registry & Metadata Services CommitteeFocus:** Design of the GDSN directory and metadata plane.

* Registry format (centralized or distributed)
* CRUD operations for namespace entries
* TTLs, leases, and heartbeat mechanisms
* Namespace indexing and searchability
* Optional features like tagging, lineage, and provenance
* Coordinates with: Backend, API, Interactions

**7.**  
**Client Tooling & SDKs CommitteeFocus:** Provide reference implementations, CLI tools, and SDKs.

* CLI tools to explore and test GDSN paths
* SDKs in Go, Python, C++, etc.
* Reference client (e.g., FUSE, browser-based explorer)
* Test harnesses and conformance suites

:repeat: Coordinates with: API, Backend**8.**  
**Governance & Compliance CommitteeFocus:** Ensure the process, licensing, and interoperability standards are enforced.

* Define versioning policy for GDSN spec
* Manage extension proposals and vendor-specific behaviors
* Ensure testability and reproducibility
* Define license (e.g., Apache 2.0) and patent policy (e.g., FRAND)
* Coordinate with SNIA, IETF, or ISO for formal adoption
* Coordinates with: All

**Bonus:**  
**Cross-Committee Working Groups**

You may want **temporary working groups** that span subcommittees to resolve tricky issues like:

* **POSIX :left_right_arrow: S3 semantic mapping**
* **Multi-protocol conflict resolution**
* **Global ACL harmonization**
* **Global cache coherency model**
* **Visualization Suggestion**

You can think of the subcommittees as working along **three planes**:  
**Layer**  
**Committees**  
**Control Plane**  
Namespace, Registry, Governance  
**Data Plane**  
Backend, API, Interactions  
**Security Plane**  
Identity & Access, Encryption, LoggingWould you like a visual org chart, a sample charter for one of the subcommittees, or a template to propose this structure to vendors or SNIA?

Absolutely. The Namespace Interactions & Referrals Subcommittee plays a critical role in defining how clients and systems navigate, discover, and resolve paths in the Global Distributed Single Namespace (GDSN). While the Backend, Namespace, and API committees define what exists and how to access it, this subcommittee governs how traversal and redirection occur across a distributed, multi-vendor namespace.

⸻

🔁 Namespace Interactions & Referrals Subcommittee

📌 Mission:

To define the protocols, mechanisms, and behaviors that allow clients and services to interact with, traverse, and resolve paths within a GDSN-aware environment, including redirection, failover, discovery, and delegation across heterogeneous storage systems.

⸻

🎯 Key Responsibilities

1. Namespace Resolution

• Define how a client resolves a GDSN path like /gdsn/orgA/eu/data/project1/file.txt to its actual backend location.

• Establish referral protocols for:

• Static mappings (predefined referral tables)

• Dynamic mappings (via namespace registry or API)

• Redirect types (HTTP-like redirect, NFS referral, or metadata-based)

2. Client-Side Referral Behavior

• Standardize how clients interpret:

• referral → remote endpoint

• redirect → different GDSN path or storage

• Define expected fallback behavior if referral fails:

• Retry with alternative endpoint?

• Cache and retry after TTL?

• Fail fast?

3. Discovery Protocols

• Mechanism for a client to "bootstrap" itself:

• Root namespace discovery

• Regional namespace discovery

• Capability advertisement (e.g., "this path supports read-only POSIX access")

• Integration with DNS, service discovery frameworks (e.g., Consul, etcd)

4. Multi-Protocol Interactions

• Define behavior when traversing between different backend types:

• From file (POSIX) to object (S3)

• From block-mapped files to virtualized objects

• How path semantics or metadata is passed between protocols

5. Namespace Traversal & Delegation

• Define how traversal works across organizational or vendor boundaries:

• Delegated authority (/gdsn/orgB is managed by a different registry)

• Sub-namespace delegation (/orgA/projects/ handled by different backends)

• Ensure delegation is secure, auditable, and fail-safe

6. Referral Formats

• Specification of how referrals are encoded:

• JSON, YAML, Protobuf

• Include TTLs, priorities, failover rules

• Metadata hints (e.g., latency, read-only, replica set)

7. Caching and TTL Management

• Define how long clients can cache resolved mappings

• Policy for:

• Cache invalidation

• Stale referral use in case of registry unavailability

• Cache warming strategies

8. Policy-Based Interaction

• Define interaction policies (which may come from the Governance committee):

• Geo-preference (e.g., prefer local replica unless stale)

• Access intent (e.g., read-intensive clients routed to specific backend)

• Compliance-based path constraints (e.g., data sovereignty)

⸻

📦 Deliverables

• Referral Protocol Specification (like a DNS SRV record but for storage paths)

• Client Referral Handling Guidelines

• GDSN Path Traversal Logic

• Referral Registry Format Specification

• Fallback and Retry Semantics

• Cross-protocol interaction behavior doc

• Test cases and mock registry services

⸻

🤝 Interfaces With

Committee Coordination Topic

Namespace To resolve hierarchical path semantics

API For traversal, lookups, and referral operations

Backend To understand actual backend mount points

Security To validate delegated access and secure redirects

Registry For registry schema, service discovery, TTL rules

⸻

🔧 Example Interaction Flow

1. User/application accesses: /gdsn/orgA/eu-west/projectX/data.csv

2. Client contacts a GDSN registry to resolve the path:

{

"path": "/gdsn/orgA/eu-west/projectX",

"referral": {

"target": "s3://orga-euwest-projectx/",

"ttl": 3600,

"access": "read-write"

}

}

3. Client follows referral to S3, accesses object

4. If failure:

• Retry alternate backend (defined in referral metadata)

• Or return error if no fallback exists

⸻

Would you like a sample JSON referral schema, a draft spec outline, or even a sample client-side behavior diagram?

Here’s a clean, **Slack-ready summary** covering all the proposed **GDSN standardization subcommittees**, ending with a deeper focus on the **Interactions & Referrals Subcommittee**.:jigsaw: **GDSN Standardization: Proposed Subcommittees Overview**We’re working toward a **Global Distributed Single Namespace (GDSN)** standard that unifies data access across multiple vendors and protocols (S3, NFS, POSIX, etc.). To structure the effort, here’s a breakdown of subcommittees:**:one: Namespace & Pathing**  
:small_blue_diamond: Define path structure (e.g., /gdsn/org/region/project/data)  
:small_blue_diamond: Handle naming collisions, hierarchy, versioning, and symlinks  
:small_blue_diamond: Align with registry and access APIs**:two: API & Access Protocols**  
:small_blue_diamond: REST/gRPC interfaces to list, resolve, and manipulate paths  
:small_blue_diamond: Support for POSIX ops, bulk traversal, metadata ops  
:small_blue_diamond: Define request/response formats and status codes**:three: Backend & Storage Integration**  
:small_blue_diamond: Map logical paths to actual storage (POSIX, S3, etc.)  
:small_blue_diamond: Support caching, consistency models, mount registration  
:small_blue_diamond: Define backend plugin/adapter architecture**:four: Identity & Access Control (Security)**  
:small_blue_diamond: Federated identity and unified ACLs (POSIX :left_right_arrow: S3 :left_right_arrow: NFSv4)  
:small_blue_diamond: Access delegation, token formats, auditing  
:small_blue_diamond: Secure traversal and namespace visibility controls**:five: Frontend Interfaces**  
:small_blue_diamond: Define how GDSN is exposed via S3, NFS, SMB, POSIX, etc.  
:small_blue_diamond: Normalize metadata, directory listing, permission behavior  
:small_blue_diamond: Advertise protocol capabilities and optimize per interface**:six: Registry & Metadata Services**  
:small_blue_diamond: Design metadata registry and discovery plane  
:small_blue_diamond: CRUD operations, TTL, leases, and indexing  
:small_blue_diamond: Namespace search and traversal metadata**:seven: Client SDKs & Tooling**  
:small_blue_diamond: CLI tools and SDKs (Go, Python, C++)  
:small_blue_diamond: Reference FUSE clients and test suites  
:small_blue_diamond: Validate cross-platform compatibility**:eight: Governance & Compliance**  
:small_blue_diamond: Handle versioning, extension rules, licensing  
:small_blue_diamond: Align with SNIA/IETF/ISO standards bodies  
:small_blue_diamond: Define conformance and vendor-neutral policy:repeat: **:nine: Interactions & Referrals (Focus Subcommittee)Mission:**  
Define how clients discover, traverse, and resolve GDSN paths across distributed systems and vendors

.**Key Topics:**  
• Path resolution (e.g., /gdsn/orgA/projectX → <s3://orgA-projX>)  
• Registry-based referrals, redirect handling, TTLs  
• Discovery protocols (bootstrapping, service discovery)  
• Failover, retry logic, and fallback behavior  
• Delegation across orgs, vendors, and regions  
• Referral format spec (JSON, Protobuf, etc.)  
• Policy-based resolution (geo, access mode, compliance)**Example Flow:**  
Client accesses /gdsn/orgA/eu-west/projectX/data.csv  
→ Registry responds with referral:

{

"referral": {

"target": "<s3://orga-euwest-projectx/>",

"ttl": 3600,

"access": "read-write"

}

}

→ Client redirects accordingly; fallback applies if S3 is down.**Coordinates With:**  
:white_check_mark: Namespace (path semantics)  
:white_check_mark: API (lookup ops)  
:white_check_mark: Security (authz for delegated paths)  
:white_check_mark: Registry (referral metadata)  
:white_check_mark: Frontend (for multi-protocol awareness)