

Data Lifecycle Management in SharePoint Online: A Framework for Policy-Based Stubbing, Retention, and Recoverability

Sreehari Prasad Jinka Venugopal

Abstract: Organizations that run collaboration on SharePoint Online accumulate content faster than they dispose of it. Storage quotas fill with files that nobody has opened in years, retention obligations are applied unevenly, and the native safety nets (recycle bins, version history and the Preservation Hold library) are widely misunderstood, which leaves recovery windows shorter than administrators assume. This paper proposes the PSRR framework (Policy-Based Stubbing, Retention and Recoverability), a lifecycle management approach that treats the three concerns as one coordinated policy set instead of three separate administrative tasks. Content is classified into lifecycle classes. Each class receives an inactivity threshold for stubbing (replacing cold content with a lightweight placeholder that keeps its location, metadata and permissions while the payload moves to a cheaper tier), a retention and disposition rule, and a recoverability tier that states how long and by which mechanism a deleted or damaged item can be restored. The framework is described against the platform behavior documented by Microsoft as of October 2023, including the two-stage recycle bin, the Preservation Hold library and Microsoft 365 Archive. To examine the design trade-offs we built a synthetic tenant of 60,000 files and ran 200 simulated replications. At a 180-day inactivity threshold, stubbing moved about 81% of stored volume to the cold tier while 24.5% of annual file accesses touched a stub, and a naive stubbing design lowered the share of accidentally deleted items that could be recovered from 82.5% to 62.3%. Coordinating payload retention with recovery windows restored this to 83.8%, and adding an independent backup layer raised it to 97.2%. These results are synthetic and show the logic of the design, not measurements from a production tenant. The paper closes with a governance model, an implementation roadmap and a research agenda.

Introduction

Data has a lifecycle whether or not an organization manages it. Files are created, used intensively for a short period, referenced occasionally, and then sit untouched for years. In an on-premises file server this pattern was visible in the storage array and was handled, if at all, by periodic clean-up projects. In SharePoint Online the pattern is hidden behind a subscription model in which storage appears elastic, sites are created freely by teams, and every Microsoft Teams team, Viva Engage community and project workspace brings its own document library. The result is rapid growth of content that is rarely opened but still counted against the tenant quota, still searchable, still indexed by assistants and still within the scope of legal and regulatory obligations.

Three practical problems follow. The first is cost and capacity: when a tenant approaches its storage quota, administrators face the choice between purchasing

MBA, Product Management

Enterprise Product Management, Data Protection, Cyber Resilience, Trustworthy AI Governance
Phoenix, Arizona, USA

mail09hari@gmail.com, ORCID: 0009-0002-5969-9357

Keywords: SharePoint Online, Data Lifecycle Management, Retention Policy, Stubbing, Archiving, Recoverability, Microsoft Purview, Records Management

capacity, deleting content, or moving content elsewhere, and each option has side effects. The second is compliance: records-management rules and privacy law require some content to be kept for defined periods and other content to be removed once its purpose has ended, so that blanket retention of everything is as problematic as aggressive deletion. The third is recoverability. Users delete files by mistake, sync clients propagate damage, ransomware encrypts libraries, and departing staff remove folders. The platform offers recovery mechanisms, but they operate on different time scales, are visible to different roles, and interact in ways that are easy to get wrong. A typical misunderstanding is that a retention policy also guarantees that a deleted item can be restored by a user; in fact, retained copies sit in a hidden library that only administrators can reach, and expired items leave that library through a staged process.

Stubbing makes these problems more intricate. A stub replaces a file with a small placeholder that preserves the name, path, metadata and permissions while the payload is stored in a cheaper tier. Done well, it reduces cost with little disruption, because users still see the file where they expect it. Done badly, it introduces new failure modes: a stub that cannot be resolved, a payload that is purged while the stub is still referenced, or a deletion that removes the placeholder but leaves the payload orphaned or, worse, removes the payload before the recovery window for the deleted stub has closed.

This paper therefore develops an integrated approach. The primary objective is to design and examine the PSRR framework. The specific tasks are: (i) to position SharePoint lifecycle management within the established phases of data lifecycle management; (ii) to document the retention, recycle-bin, archive and backup mechanisms that the framework builds upon and the gaps between them; (iii) to define lifecycle classes and a policy matrix that couples stubbing thresholds, retention rules and recoverability tiers; (iv) to specify safeguards that keep stubs, payloads and recovery windows consistent; and (v) to examine the resulting trade-offs in a controlled simulation. The contribution is a coherent blueprint that administrators, records managers and security teams can adopt together, together with explicit statements about what the evidence does and does not show.

The paper is organized as follows. Section II reviews the literature. Section III summarizes related work. Section IV describes the lifecycle mechanisms of SharePoint Online. Section V presents the PSRR framework. Section VI details the experimental setup, Section VII reports results, Section VIII discusses implications, Section IX addresses limitations, and Section X concludes.

Literature Review

A. Data lifecycle management

Data lifecycle management (DLM) is the practice of governing information from its creation to its final disposition. Figure 1 shows a widely used five-phase view: creation and capture, storage and organization, usage and sharing, archiving and retention, and deletion and disposal. The phases are sequential for any single item, but in an organization they overlap, because content at every phase exists at the same time and policies must be evaluated continuously.

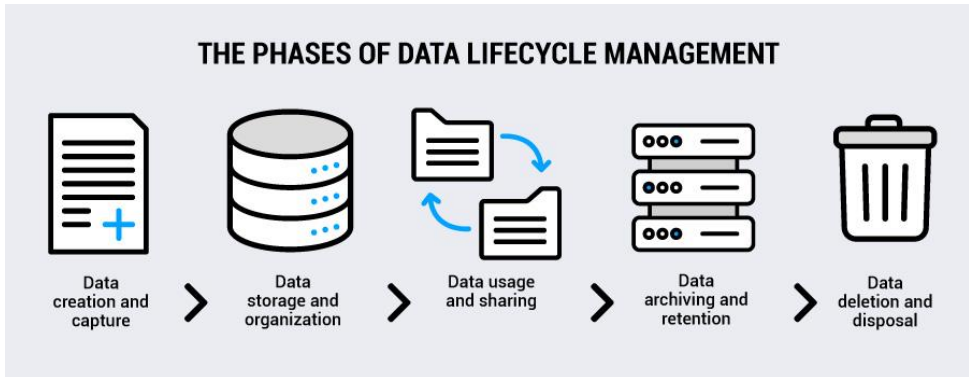


Figure 1. The phases of data lifecycle management: creation and capture, storage and organization, usage and sharing, archiving and retention, and deletion and disposal.

The idea of managing storage according to the changing value of information has a long history. Chen argued that information valuation should drive information lifecycle management, placing data on storage tiers that match its current business value [1]. Empirical file-system studies support the premise. Leung and colleagues analyzed workloads in large enterprise network file systems and observed that most files are not reopened after creation and that the set of files in active use is a small fraction of what is stored [2]. Meyer and Bolosky's study of deduplication in desktop file systems likewise found large amounts of redundant and stale content [3]. These observations explain why tiering produces savings, and also why a small share of cold files will nevertheless be needed again, which is the central risk of any stubbing policy.

B. Retention, records and disposition

Retention is the other half of the lifecycle. ISO 15489-1 sets out principles for records management, including that records must be authentic, reliable, complete and usable for as long as they are required, and that disposition must be authorized and documented [4]. Regulation adds specific obligations. The European General Data Protection Regulation requires that personal data be kept no longer than necessary for its purpose, which makes disposal a legal duty as well as a cost measure [5]. Financial regulation, such as the United States SEC rules on broker-dealer records, requires certain records to be preserved in non-rewritable form for defined periods [6]. For organizations subject to both kinds of rule, retention is not simply "keep everything", and a rule that is too long can be as non-compliant as one that is too short.

Disposal also has a security dimension. NIST Special Publication 800-88 provides guidelines for media sanitization and stresses that deletion must be matched to the sensitivity of the data and verified [7]. For a cloud service the customer cannot sanitize media directly, so the practical controls are policy configuration, auditing and the contractual and technical assurances of the provider.

C. Backup, versioning and recovery

The recovery literature distinguishes between mechanisms that protect against different failure types. Chervenak and colleagues surveyed backup techniques and noted that snapshots, logs and full copies differ in what they protect, how quickly they restore and how long they keep history [8]. Santry and colleagues' Elephant file

system explored the question of when a system should forget old versions, proposing policies based on the value of versions and not simply on age [9]. The lesson for SharePoint is that version history, recycle bins and backups are complementary: version history protects against unwanted edits, recycle bins protect against deletion for a limited time, and backup protects against broader loss, including malicious and systemic events, over a longer horizon.

D. Tiering and stubbing

Hierarchical storage management has traditionally used stubs, sometimes called placeholders or shortcuts, to leave a visible pointer in the file system when content is moved to secondary storage. The benefit is transparency to users and applications. The costs are latency on first access, dependency on the secondary store, and the need to keep pointer and payload consistent through copy, move, delete, restore and migration operations. In SharePoint Online, the equivalent native concept is Microsoft 365 Archive, which moves content to a cold storage tier inside the Microsoft 365 boundary. It was introduced for entire sites, and file-level archive has since extended the approach to individual files [10]. Archived content remains subject to the tenant's security and compliance configuration, and reactivation is requested explicitly and can take hours and not seconds. Third-party products use a different pattern in which a placeholder in SharePoint points to a payload in an external repository. This paper uses the word "stubbing" for the general pattern, independent of whether the tier is native or external.

Related Work

Prior work approaches the topic from storage engineering, records management and platform administration, and few contributions connect all three.

In storage engineering, hierarchical storage management and information lifecycle management literature focuses on cost-effective placement of data on tiers according to access frequency and value [1], [2]. These studies offer models for deciding when to move data but rarely examine compliance interactions: they assume that a file that is moved can later be deleted by the same system, whereas in a records-managed environment deletion is subject to retention rules, holds and audit.

In records management, standards and practitioner guidance describe retention schedules, classification schemes, disposition review and legal holds [4]. These sources are detailed about what must be retained and for how long, and less detailed about technical realization in a particular collaboration platform, for example how a retention label interacts with a recycle bin or with an archive tier.

In platform administration, Microsoft's documentation, community guidance and consultancy articles describe the Preservation Hold library, retention policy and label behavior, the two-stage recycle bin, versioning limits, the archive service and backup offerings [10], [11]. This body of knowledge is practical and frequently updated. Its limitation is fragmentation: each feature is explained on its own, and the combined behavior, particularly at the boundaries, has to be inferred by the administrator. For example, the documentation notes that the recycle bin is not indexed and therefore cannot be searched by eDiscovery, that expired content from the Preservation Hold library now passes through the second-stage recycle bin and that the clean-up job runs weekly and considers only content older than

30 days [11]. Each statement is correct, but together they determine an effective recovery window that few organizations have calculated.

Finally, work on resilience in cloud collaboration emphasizes that provider-side durability does not equal customer-side recoverability. A provider may store several copies of data to protect against hardware failure, yet a user-initiated deletion or a malicious encryption event is, from the platform's perspective, a legitimate change that is replicated faithfully. This is the reason independent backup remains part of good practice, and it informs the recoverability tiers in PSRR.

The gap that PSRR addresses is therefore one of integration. No widely adopted approach (a) defines lifecycle classes that drive stubbing, retention and recovery from a single policy matrix, (b) states the effective recovery window of each class in plain terms, (c) prevents inconsistencies between stubs and payloads, and (d) provides a governance model with review and audit.

Lifecycle Mechanisms in Sharepoint Online

This section summarizes the platform behavior that PSRR relies on. Statements follow Microsoft documentation as consulted in October 2023; because retention and archive behavior has changed several times, administrators should verify details against current documentation before implementation.

A. Retention policies, labels and the Preservation Hold library

Retention in Microsoft Purview is configured through retention policies, which apply to locations such as sites, and retention labels, which apply to individual items and can be applied manually, by rule or by classifier. A setting can retain content, delete it after a period, or both. To support retention, SharePoint creates a hidden Preservation Hold library in a site when needed. When content under retention is modified or deleted during the retention period, a copy of the original is created in that library [11]. The library is not visible to end users, so recovery from it is an administrative action.

A timer job evaluates the library periodically. It considers only content that has been in the library for more than 30 days and runs every seven days, so that, after the 30-day minimum, it can take up to 37 days for expired content to be removed from the library [11]. Removal is no longer a permanent deletion: content that has outlived its retention is moved to the second-stage recycle bin, where it is permanently deleted at the end of 93 days [11]. If the content was not changed or deleted during the retention period, the timer job moves it to the first-stage recycle bin at the end of that period, and the 93-day window spans both stages.

B. The two-stage recycle bin

When a user deletes a file, it moves to the first-stage (user) recycle bin, from which the user can restore it. If the user deletes it from there or empties the bin, it moves to the second-stage (site collection) recycle bin, which administrators can see. The 93-day period spans both stages and starts at the original deletion, and the recycle bin is not indexed, so its content cannot be found by search or placed on hold by eDiscovery [11]. Microsoft does not publish a tenant-configurable duration for this bin, so the effective recovery window for an ordinary deletion in a site without retention is fixed at 93 days.

C. Figure 2: the two paths

Figure 2 summarizes the two paths that a document can take. On path 1, a retention policy applies, and a modified or deleted item is preserved in the Preservation Hold library until retention ends, after which a clean-up job removes it. On path 2, a user deletes an item, which passes through the first-stage recycle bin, a user purge to the second-stage bin, and clean-up, with permanent deletion at 93 days in total. The figure shows the seven-day clean-up cycle on path 1 leading straight to permanent deletion. Current documentation adds a step: items leaving the Preservation Hold library are placed in the second-stage recycle bin and are permanently deleted only after a further 93 days. We keep the simplified figure because it conveys the two independent time scales, and the framework below uses the documented, more conservative behavior.

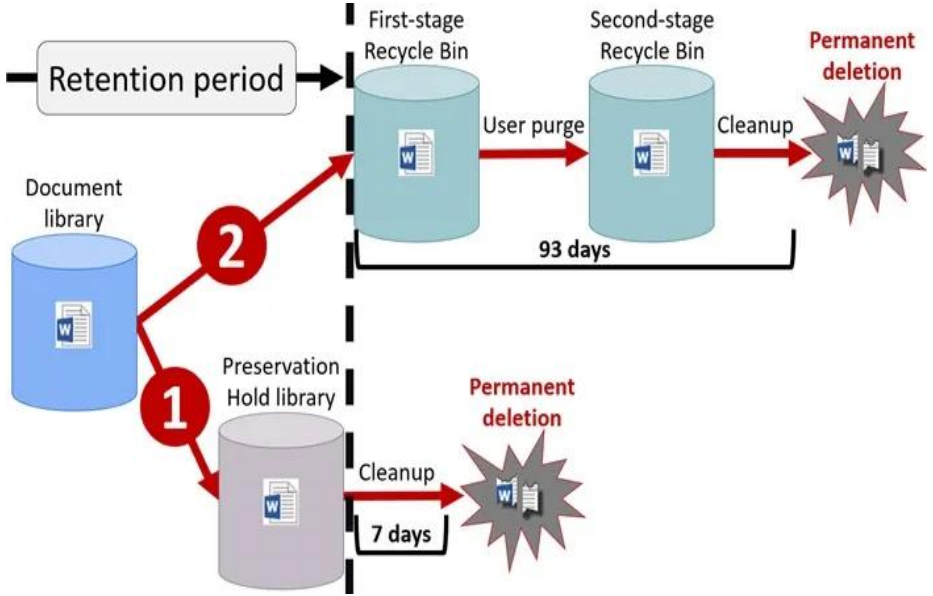


Figure 2. Retention and deletion paths for a document library item: preservation hold (path 1) and recycle bins (path 2), with the seven-day clean-up cycle and the 93-day recycle-bin period.

D. Versioning, archive and backup

Version history keeps earlier states of a file and protects against unwanted edits, but it consumes storage and is bounded by version limits, and under retention the Preservation Hold library may keep additional copies. Microsoft 365 Archive moves inactive sites, and more recently individual files, to a cold tier while preserving metadata, permissions, search indexing and compliance features; reactivation is a request that can take up to 24 hours [10]. Microsoft 365 Backup and third-party backup services provide point-in-time restore across a longer horizon than the recycle bin. Table 1 summarizes the mechanisms and their limits as used in this paper.

TABLE 1. Native Lifecycle Mechanisms and Their Effective Windows

Mechanism	What it protects against	Effective window	Main limitation
Recycle bin (two stages)	Accidental deletion	93 days from deletion	Not indexed; not searchable or holdable
Preservation Hold library	Edits and deletions of content under retention	Retention period, plus up to 37 days, plus 93 days in recycle bin	Hidden; administrator recovery; adds storage
Version history	Unwanted edits	Until version limits or retention rules apply	Consumes storage; does not protect against site loss
Microsoft 365 Archive	Cost of inactive content	Indefinite while archived	Reactivation can take up to 24 hours
Backup (native or third-party)	Broad loss, malicious change	Depends on service and configuration	Separate service; separate cost and testing

Methodology: The PSRR Framework

A. Design principles

PSRR is guided by five principles. First, *one policy matrix*: stubbing, retention and recovery are decided together for each lifecycle class so that their effects are never contradictory. Second, *evidence before action*: a policy is applied only after usage and classification data show what it will affect, and it is piloted before it is widened. Third, *transparency to users*: stubbing must not break links, permissions or search, and users must be told what a placeholder means and what to expect on first access. Fourth, *no silent loss*: no step in the lifecycle may reduce a recovery window without that reduction being explicit in the policy matrix. Fifth, *independent recoverability*: native features are supplemented by a separate backup so that a single administrative error or compromise cannot remove every copy.

B. Lifecycle classes

Content is assigned to classes by site purpose, sensitivity and regulatory requirement. Class assignment can be automated by site templates and sensitivity labels and refined by retention labels at item level. An illustrative set is shown in Table 2. The classes are examples, and each organization should replace them with its own records schedule.

TABLE 2. Illustrative Policy Matrix (values are examples to be set by each organization)

Lifecycle class	Typical content	Stub after inactivity	Retention and disposition	Recoverability tier
Transient	Meeting notes, drafts, temporary working files	90 days	Delete after 2 to 3 years	Recycle bin and backup
Operational	Project documents, team collaboration	180 days	Retain 5 years, then review	Recycle bin, versions and backup
Regulated records	Contracts, financial and HR records	365 days, or never when regulation requires	Retain per schedule, then disposition review with approval	Preservation hold, backup and restore testing
Reference	Policies, templates, finished deliverables	Not stubbed	Retain until superseded	Recycle bin, versions and backup
Hold	Content under legal hold	Not stubbed unless counsel approves	Retain until hold is released	Preservation hold and backup

C. Stubbing design

A stub in PSRR is a placeholder that keeps the name, location, metadata, permissions, sharing links and version chain of the original, while the payload moves to a cold tier. Four rules govern it.

Eligibility. An item is eligible when it has been inactive for the threshold of its class, is larger than a minimum size below which stubbing saves little, and is not under a hold or in a class that excludes stubbing. Inactivity is measured from the last access or modification recorded in audit data and not from creation date, because many valuable items are old but regularly consulted. A short list of exemptions protects content with known recurring use, such as audit evidence and templates.

Transparency. Placeholders must resolve through the original URL so that existing links continue to work. Users are told, through a visible marker and through guidance, that first access may be slower. Where the native archive tier is used, the documentation states that reactivation can take up to 24 hours, so critical content should not be stubbed to a tier whose recall time is longer than the business can tolerate.

Recall. Recall is a managed event. The system logs it, reports it, and uses it as feedback: a class whose recall rate is high indicates a threshold that is too short. Recalled items are returned to the active tier and restart their inactivity period.

Integrity. A stub and its payload form a unit. Operations that affect one must affect the other consistently. Copy and move preserve both; deletion of a stub starts the recoverability

clock for the payload, which must remain restorable for at least as long as the stub itself; and migration or tenant restructuring must carry stubs and payloads together. These rules address the main hazards of stubbing, and they are tested explicitly in the simulation below.

D. Retention and disposition design

Retention in PSRR is declarative. Each class has a retention label or policy that states what is kept, for how long, and what happens at the end. Several decisions deserve attention.

Retain versus delete. Where regulation requires preservation, use retain-and-delete or retain-only rules. Where privacy law drives disposal, use delete rules with a defined review step. For content with no obligation, short retention is preferable because every retained copy also counts toward storage and exposure.

Disposition review. For regulated classes, end-of-retention deletion is preceded by a review in which a records owner can approve, extend or exempt items, and the outcome is logged.

Preservation Hold library planning. Because modifications and deletions under retention create copies, storage consumption rises on retained sites. PSRR budgets for this by monitoring the size of the library on affected sites and including it in capacity planning, and by scoping retention policies to the locations that need them instead of applying them tenant-wide by default.

Disposal timing. The administrator should state the full time from the end of retention to permanent deletion, which includes the weekly timer job, the 30-day minimum and the 93-day recycle-bin period. Where privacy law requires erasure on request, this lag must be considered when promising a deletion date.

E. Recoverability design

Recoverability is expressed as tiers that define who can restore, from where, and until when.

Tier 1: user self-service. The first-stage recycle bin and version history allow users to undo mistakes without help. The window is 93 days for deletion, subject to the platform behavior in Section IV.

Tier 2: administrator restore. The second-stage recycle bin and, for content under retention, the Preservation Hold library allow administrators to restore content that users can no longer see.

Tier 3: independent backup. A backup service with point-in-time restore across a longer horizon, such as a year, protects against delayed discovery, large-scale deletion and malicious change. Backup must be tested: a restore exercise should be run at least annually for regulated classes and its result recorded.

Tier 4: legal and forensic preservation. Holds and eDiscovery processes preserve content for investigations and are separate from operational recovery.

The policy matrix assigns each class to the tiers it needs and states the resulting effective window. Two rules link recoverability to stubbing. A payload belonging to a deleted stub must be held for at least the longest recovery window assigned to the class. And restore tests must include stubbed content, since a restore that returns placeholders without payloads has not restored the file.

F. Governance and operating model

PSRR assigns roles. A *records owner* maintains the retention schedule and approves dispositions. A *platform owner* operates policies, archive and backup. A *security owner* monitors deletion events, unusual bulk activity and access to hidden libraries. *Site owners* confirm class assignment and exemptions. A *lifecycle board* meets quarterly to review recall rates, storage trends, restore test results and exceptions.

Measurement is part of governance. We recommend tracking the share of stored volume in each tier, the recall rate and recall time for stubs, the size of preservation libraries, the share of content past its retention that has not yet been disposed of, the proportion of restore tests that succeeded, and the number of policy exceptions. These indicators replace opinion with evidence when thresholds are reviewed.

G. Implementation roadmap

The roadmap has four stages. In *discovery*, the organization inventories sites, sizes, owners, sensitivity and activity, and identifies the regulated classes. In *design*, it builds the policy matrix and agrees thresholds with owners. In *pilot*, it applies policies to a small group of low-risk sites, measures recall and restore behavior, and adjusts. In *rollout*, it extends policies in waves, with communication to users, and moves into steady-state governance. Each wave has an explicit rollback plan, because both stubbing and retention changes can be hard to reverse once their effects have propagated.

Experimental Setup

The experiments are limited in purpose. A production tenant cannot be used to compare design choices safely, and no organization can observe the counterfactual of a design it did not adopt. We therefore used a synthetic tenant to examine three questions: how much volume stubbing can move and how often stubs are touched, how much content becomes eligible for disposition under a retention schedule, and how the combination of stubbing and recovery design affects the share of accidentally deleted items that can be restored. The simulation was written in Python with a fixed random seed. Each reported figure is the mean of 200 independent replications.

Synthetic tenant. Each replication contains 60,000 files. File age is drawn from a distribution weighted toward recent content, with ages up to ten years, and file size follows a log-normal distribution with a median of 0.8 MB. Idle time, the number of days since the last access, is a random fraction of file age, so that older files tend to be idle longer. A file's probability of being accessed during the next year depends on its idle time: high for files idle for less than 90 days, then falling in steps to a very low value for files idle for more than two years. A subgroup of 5% of files is treated as dormant but valuable, with a floor on its access probability to represent audit evidence and reference material that is rarely opened but important.

Stubbing experiment. Files are eligible if they exceed 0.25 MB and have been idle for at least a threshold of 90, 180 or 365 days. For each threshold we record the share of files stubbed, the share of stored volume moved to the cold tier, the number of stub recalls per 1,000 files per year, and the share of all annual accesses that touch a stub.

Retention experiment. Each file is assigned to a retention class (three years, five years, seven years or ten years) with fixed class proportions, and 3% of files are exempted to represent legal holds. We record the share of files and of volume that is older than its retention period and therefore eligible for disposition review.

Recoverability experiment. In each replication, 2,000 files are deleted by mistake, with deletion more likely for older files. The delay before the deletion is discovered follows a log-normal distribution with a median of 25 days and a long tail. A deleted file is recoverable if it is discovered within the window of the design under test. Four designs are compared: (a) native only, with the 93-day recycle bin window; (b) naive stubbing, in which the payload of a deleted stub is cleaned up after 30 days while non-stubbed files keep the native window; (c) PSRR without independent backup, in which payloads are held for 100 days so that they outlast the recycle bin; and (d) PSRR with a backup layer whose restore horizon is one year.

All assumptions, including the shape of the access distribution, the discovery-delay distribution and the 30-day payload clean-up, are inputs of the simulation and not measurements.

Results

A. Stubbing: savings and recall

Table 3 reports the stubbing experiment. A threshold of 180 days placed about 65% of files and about 81% of stored volume on the cold tier, which shows the large savings available in a tenant with many old, large files. It also meant that about 43 stub recalls per 1,000 files occurred each year, and that about one in four annual accesses touched a stub. A 90-day threshold moved nearly 89% of volume but made more than 40% of accesses recalls, while a 365-day threshold reduced recalls to about 14% of accesses at the price of a smaller saving (about 67% of volume).

TABLE 3. Simulated Effect of the Stubbing Threshold (60,000 files, 200 replications)

Inactivity threshold	Files stubbed (%)	Volume on cold tier (%)	Recalls per 1,000 files per year	Share of annual accesses that are recalls (%)
90 days	71.9	89.0	72.8	41.6
180 days	65.2	80.8	43.0	24.5
365 days	53.7	66.5	24.9	14.2

The pattern is typical of tiering problems: most of the savings come from a small number of large, old files, so a longer threshold surrenders relatively little saving and removes a large share of recalls. The 180-day setting is a compromise and not an optimum. The dormant-but-valuable subgroup contributes disproportionately to recalls, which is the reason for the exemption list in the framework.

B. Retention: disposition eligibility

Under the illustrative schedule, about 33.4% of files, representing about 33.5% of stored volume, were older than their retention period and not on hold, and therefore eligible for disposition review. In other words, a disciplined schedule can make a third of the tenant a candidate for removal, which is a larger saving than stubbing alone offers for content that has no continuing value. The result also illustrates the order of operations: items that are eligible for disposition should be reviewed before they are stubbed, because paying to archive content that is about to be deleted is waste.

C. Recoverability

Table 4 reports the share of accidentally deleted files that could be recovered under each design. With the native 93-day window, 82.5% of deleted files were recoverable; the rest

were discovered too late. Naive stubbing reduced this to 62.3%, because about three quarters of deleted files in this population were stubs (deletion was weighted toward older, colder content) and their payloads were cleaned up after 30 days, which left only 55.2% of deleted stubs recoverable. Holding payloads for 100 days removed this penalty and brought recoverability to 83.8%, a result slightly better than the native figure only because the payload hold is a little longer than the recycle bin. The largest improvement came from adding an independent backup with a one-year horizon, which raised recoverability to 97.2%.

TABLE 4. Simulated Recoverability of Accidentally Deleted Files (2,000 deletions per replication)

Design	Recoverable (%)
(a) Native only, 93-day recycle bin	82.5
(b) Naive stubbing, 30-day payload clean-up	62.3
(c) PSRR without backup, 100-day payload hold	83.8
(d) PSRR with one-year backup	97.2

D. Summary of findings

Three findings stand out. First, stubbing yields large savings, but its benefit depends strongly on the threshold, and recall frequency rises quickly as thresholds shorten. Second, a stubbing design that is not coordinated with recovery windows can reduce recoverability by about twenty percentage points, even though no individual component misbehaves. Third, the remaining unrecoverable cases in a recycle-bin-only design are those discovered late, and only a longer-horizon mechanism such as backup closes that gap. Coordination prevents regression, but independent backup creates improvement.

Discussion

The simulation supports the central argument of the framework: stubbing, retention and recoverability interact, and should be designed together. The most important interaction is the one that the naive design exposed. Stubbing changes where a payload lives and, unless the payload's retention is tied to the recovery promise made for the deleted item, the recovery promise silently shortens. This is a design hazard and not an implementation detail, so it belongs in the policy matrix and in testing.

The results also clarify the roles of the native mechanisms. The recycle bin is a user safety net with a fixed, relatively short window. The Preservation Hold library is a compliance mechanism, not a convenience, and its administrator-only access, its storage cost and its staged exit make it unsuitable as the primary recovery tool for ordinary users. Microsoft 365 Archive reduces cost without leaving the compliance boundary but introduces a recall time that some processes cannot tolerate. Backup fills the longer horizon. A lifecycle that uses each for its proper purpose is both cheaper and safer than one that asks a single feature to do everything.

For practitioners, four recommendations follow. Begin with discovery and measurement, because a policy tuned to real inactivity data will outperform a policy tuned to intuition. Apply disposition before archiving, so that content with no remaining value is not paid for twice. State recovery windows per class in plain language, including the staged exits and weekly job cycles that make real windows longer than the headline numbers. And test restores, including restores of stubbed items, because an untested recovery capability is an assumption.

The framework also has implications for compliance. Where regulation requires that records remain immutable or preserved for specified periods, the policy matrix should record which mechanism provides the preservation and how it has been verified, so that an audit can follow the chain from rule to configuration to evidence. Where privacy law requires erasure, the matrix should state how long the platform takes to complete deletion, including recycle-bin and backup retention, so that statements made to individuals are accurate.

Finally, the framework is deliberately conservative about automation. Retention and disposal decisions have legal consequences, and stubbing decisions affect user experience. Automated classification and policy assignment are useful, but exemptions, disposition approvals and threshold changes remain human decisions with logged rationale.

Limitations and Future Work

The principal limitation is that the evaluation is synthetic. The access probabilities, the age and size distributions, the deletion pattern and the discovery-delay distribution are assumptions chosen to be plausible and were not fitted to a real tenant. Absolute numbers, such as the share of volume moved or the recall rate, will differ in practice and could be materially different in tenants dominated by large media files, by many small documents, or by heavily collaborative content. The relative conclusions, especially the cost of uncoordinated stubbing and the gap closed by backup, follow largely from the structure of the design and are likely to be more robust than the levels, but they have not been confirmed in the field. The experiments also did not vary the discovery-delay distribution or the access model in a sensitivity analysis.

Platform behavior is a second source of uncertainty. Retention, archive and backup features have changed repeatedly, and documentation as of October 2023 may be superseded. Recall times, limits and pricing for archive tiers should be verified, and the third-party stubbing tools that follow a different pattern from the native archive were not evaluated.

Third, the framework addresses documents in SharePoint document libraries. Lists, pages, Teams chat content, Loop components and mailboxes follow different rules and are outside scope.

Future work should therefore include: (i) a pilot in a real tenant with prospective measurement of recall, restore success and storage; (ii) sensitivity analyses over access and discovery-delay models; (iii) comparison of native and external stubbing designs with respect to integrity and recall time; (iv) study of the interaction between retention labels, version limits and Preservation Hold library growth; and (v) extension of the policy matrix to non-document content and to ransomware scenarios with realistic attacker behavior.

Conclusion

This paper presented PSRR, a framework for data lifecycle management in SharePoint Online that combines policy-based stubbing, retention and recoverability in a single policy matrix. It positioned the problem within the phases of data lifecycle management, described the retention, recycle-bin, archive and backup mechanisms and their effective windows, and set out lifecycle classes, stubbing rules, retention and disposition rules, recovery tiers and a governance and implementation model. In a controlled simulation, stubbing at a 180-day threshold moved about 81% of volume to a cold tier with about one quarter of accesses touching a stub; a naive stubbing design lowered recoverability of accidental deletions from 82.5% to 62.3%; coordinating payload retention with recovery windows restored it to 83.8%; and an independent backup raised it to 97.2%.

The simulation is evidence of coherence and not of field effectiveness. The next step is a pilot against a real tenant with measured recall, restore and storage outcomes. If those results are consistent with the logic shown here, organizations can lower storage cost, meet retention obligations and keep a credible recovery promise, provided they treat the three as one design problem.

References

- [1] Y. Chen, “Information valuation for information lifecycle management,” in Proc. 2nd Int. Conf. Autonomic Computing (ICAC), 2005, pp. 135–146.
- [2] A. W. Leung, S. Pasupathy, G. Goodson, and E. L. Miller, “Measurement and analysis of large-scale network file system workloads,” in Proc. USENIX Annual Technical Conf., 2008, pp. 213–226.
- [3] D. T. Meyer and W. J. Bolosky, “A study of practical deduplication,” in Proc. 9th USENIX Conf. File and Storage Technologies (FAST), 2011, pp. 1–13.
- [4] International Organization for Standardization, ISO 15489-1:2016 Information and documentation: Records management, Part 1: Concepts and principles. Geneva, Switzerland: ISO, 2016.
- [5] European Parliament and Council, “Regulation (EU) 2016/679 (General Data Protection Regulation),” Official Journal of the European Union, L 119, 2016.
- [6] U.S. Securities and Exchange Commission, “Rule 17a-4: Records to be preserved by certain exchange members, brokers and dealers,” 17 C.F.R. § 240.17a-4.
- [7] R. Kissel, A. Regenscheid, M. Scholl, and K. Stine, Guidelines for Media Sanitization, NIST Special Publication 800-88, Revision 1, National Institute of Standards and Technology, 2014.
- [8] A. L. Chervenak, V. Vellanki, and Z. Kurmas, “Protecting file systems: A survey of backup techniques,” in Proc. Joint NASA/IEEE Mass Storage Systems Conf., 1998.
- [9] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir, “Deciding when to forget in the Elephant file system,” in Proc. 17th ACM Symp. Operating Systems Principles (SOSP), 1999, pp. 110–123.
- [10] Microsoft, “Microsoft 365 Archive,” Microsoft Learn documentation, <https://learn.microsoft.com/en-us/microsoft-365/archive/archive-setup>, accessed Oct. 2023.
- [11] Microsoft, “Learn about retention for SharePoint and OneDrive,” Microsoft Purview documentation, <https://learn.microsoft.com/en-us/purview/retention-policies-sharepoint>, accessed Oct. 2023.