IT Project Guidance

On the removal of Personal Information from Systems

Version: 0.4

Purpose

This document provides authoritative guidance for government projects on the appropriate handling of personal information (PI) within information systems. It addresses entrenched assumptions that deletion implies physical removal and outlines practical, regulatory-compliant alternatives. This guidance supports decision-makers, business analysts, architects, and developers in designing systems that balance privacy, accountability, operational resilience, and regulatory obligations.

Synopsis

Systems handling personal information must balance privacy protection with auditability, accountability, and operational integrity. While some policies and regulatory interpretations have equated deletion with physical removal, this view is increasingly unworkable. Deletion, in the architectural sense, should be understood as a transition in the information lifecycle—often to a non-identifiable or anonymised state. This paper demonstrates why physical deletion is usually undesirable and unnecessary, introduces a domain-aware alternative grounded in state transitions and versioning, and provides both conceptual and technical guidance to implement these principles in new and existing systems.

Contents

Introduction

Personal information is deeply embedded in the operation of public systems—from education and health to identity and social support. Data about people is retained not only for service delivery, but for accountability, longitudinal analysis, and rights protection. Yet the act of "deletion" has been mischaracterised as a moral or legal imperative—something the system must perform to meet obligations of privacy. That assumption has become an error.

Deletion originally addressed a hardware constraint: the physical removal of data to make space. It became a software habit. Then, a legal overreaction. With the introduction of the GDPR and NZ Privacy Act 2020, deletion was reframed as ethical good practice—but without clarifying that the requirement was not always for physical erasure. The result was panic-driven development: erasing records without preserving meaning, continuity, or accountability.

This document counters that misunderstanding. It explains how systems can respect the right to be forgotten while retaining their obligation to remember. It makes the case for functional removal over physical deletion, and it shows how architectures based on state, not destruction, yield more resilient, auditable, and maintainable systems.

Background

Record deletion began as a cost-saving mechanism. Storage was expensive and limited. Data deemed old or inactive was purged or moved to tape. This mirrored physical filing: shifting paper archives to storage sheds.

But when data storage became cheap, deletion remained. Now detached from its original purpose, it was applied uncritically to regulatory compliance. GDPR-era panic created a compliance industry focused on deletion-as-virtue—even when it undermined audit trails and operational reporting.

Compounding this shift was the elevation—and misapplication—of the CRUD pattern: Create, Read, Update, Delete. Originally a way to describe low-level operations that a relational database could perform, CRUD was never intended to model the lifecycle of meaningful information. Nor was it sufficient for regulatory compliance, state tracking, or the complexities of privacy management. Despite this, it came to dominate not only database logic, but also API patterns and user interface design.

RESTful APIs in particular were frequently described—mistakenly—as being based on CRUD. This misunderstanding stems from their shared verbs (POST, GET, PUT, DELETE), but the similarity is superficial. REST is a stateless transport protocol model with no opinion on lifecycle, purpose, or meaning. Its verbs facilitate interaction, but they do not constrain it to data-centric thinking. When REST is used merely to implement CRUD, the system inherits the same design limitations: oversimplified transitions,

irreversible deletions, and poor alignment with the real-world states that information moves through.

This was not merely an overextension—it was a category error. It reflected a shift from working with information to working with data. Data concerns storage and manipulation—how facts are captured, stored, and retrieved. Information, by contrast, carries intent and meaning; it exists to support decision-making, accountability, and purposeful action. The former is a technical asset, but when misapplied, can become debt. The latter is a social and operational asset, capable of influencing behaviour and delivering public value. Systems built on CRUD manage data, but they do not steward information. They optimise for records, not relevance.

Information management involves far more than four basic operations. Records are created, reviewed, amended, versioned, retracted, archived, anonymised, and occasionally destroyed. The lifecycle of information, especially when personal, is shaped by purpose, context, and consent. Yet under CRUD, all of this nuance was collapsed into oversimplified functions, particularly the irreversible and poorly understood 'Delete.'

Most damaging was the assumption that 'Delete' in CRUD, when used to manage information, equated to legal or operational removal. This overlooked that deletion is a physical act, not a legal necessity. True system design must distinguish between functional state transitions (such as deactivation, redaction, or anonymisation) and the irreversible destruction of data. By framing deletion as a default operation, CRUD-centric thinking embedded fragility, obscured state management, and led to irreversible data loss under the false flag of compliance.

Consider education data. If students over a certain age are deleted in line with poorly interpreted policy, longitudinal reports show classrooms of zero students. This is not privacy—it is institutional amnesia. Worse, deletion creates analytic distortions and undermines accountability. Records of consent disappear. Histories collapse.

An industry of data warehouse tier based retrospective analytics emerged to fill the operational void left by aggressive deletion. These systems promised long-term insight, but did so by extracting data from its original context. Data warehouses typically contain sanitised, standardised versions of operational data, often delayed, and decoupled from the systems of action. As a result, while they support broad pattern analysis or policy research, they are ineffective at informing real-time decisions, issue triage, or service improvement at the point of need.

This separation introduces latency—not just technical, but institutional. Access to the warehouse becomes governed by process, request queues, and reporting cycles. The result is diminished immediacy, and diminished operational value. Analytical insight, once actionable, becomes archaeological. In contrast, operational data hubs—governed, queryable, identity-aware—preserve context and enable secure, timely access. These offer a far better foundation for real-time insight without compromising auditability or privacy.

Anonymisation satisfies the legal test, and it preserves the value of what was recorded.

Context

Many systems in government and enterprise still operate under outdated assumptions about the nature and necessity of deletion. These assumptions are not just technical defaults—they are baked into institutional thinking, procurement language, and development practices. The result is a design pattern that treats deletion as mandatory, irreversible, and protective, when in fact it may be none of those things.

Developers and policy writers often assume that physical deletion is legally required, that it is safe and reversible, that it minimises liability, and that it simplifies data governance. In reality, the inverse is often true. Anonymisation or simple removal of access usually satisfies privacy obligations. Physical deletion, by contrast, breaks audit trails, conceals decision history, and undermines operational accountability. Once deleted, there is no recovery—no way to reconstruct what occurred, who acted, or why a service pathway was followed.

Additionally, technical constraints further complicate matters. Personal information is rarely confined to a single table or field. Identifiers often appear in free text—notes, forms, comments—or in unstructured logs. Many systems lack proper version control or structured redaction capabilities. SaaS platforms, particularly those designed for commercial enterprise rather than public accountability, may provide no support for audit-compliant redaction or controlled anonymisation.

These problems converge operationally. Users become hesitant to act. They fear triggering irreversible deletions. Modal confirmation dialogues become widespread—not as features, but as band-aids over brittle logic. This inflates training time, reduces confidence, and slows decision-making across roles.

Understanding the assumptions and architectural patterns that gave rise to deletion-centric thinking is essential for reform. The remaining sections of this guidance address how to move beyond deletion—towards design models grounded in traceability, statefulness, and legal sufficiency through anonymisation.

Problem Visualisation

Imagine personal information as bricks in a load-bearing wall. Each brick has a purpose, place, and holds a detail—name, address, diagnosis—but the wall itself represents trust: a structured, traceable narrative of service and accountability. Remove one brick, and the wall might still stand. Remove many, and the structure weakens, collapses, and loses meaning. This is what happens when deletion is treated as physical removal.

Instead, records should be preserved structurally while removing identifier details.

Replace the bricks, not the wall. Substitute identity with anonymised placeholders. The wall remains. The load is still carried. And trust—built on continuity and evidence—is maintained.

Assumptions in Current Thinking

Several assumptions continue to shape system requirements and stakeholder expectations:

1. Deletion is mandatory: Most assume regulations require physical deletion of all PI after expiry or request.
2. Deletion is safe: Stakeholders believe data can be removed without impact to reference integrity or future service.
3. Deletion reduces liability: It is assumed that once PI is deleted, the organisation is protected.
4. Systems can delete effectively: It is assumed developers can reliably identify and remove all personal data.
5. SaaS tools are compliant: Agencies believe commercial software automatically implements best-practice deletion.

These assumptions do not hold in practice. Legal provisions allow for retention of data under legitimate interest, accountability, or consent withdrawal tracking. Systems frequently store personal data across tables, logs, and external services. Removal may create orphaned records or break relational links. SaaS products often lack transparency or redaction tooling. Worse, the absence of data disables support staff from verifying what occurred—leading to service degradation.

Problems with Deletion

- Deletes audit trails and distorts records
- Removes data needed for support or analysis
- Hides decision rationale
- Introduces inconsistency between systems
- Is often irreversible
- Encourages modal confirmations, slowing operations

Worse, deletion creates fear. Modal dialogues arise as a patch. "Are you sure you want to delete?" is a sign of design failure. It reveals that actions are irreversible. Tentability sets in.

When systems are reversible, modals vanish. Confidence grows. Training costs drop. Staff work without fear.

Clarifying Mandated Action Terms

Because this subject arises from obligations in law and regulation, it is essential to begin with a shared and precise understanding of key terms. Misuse or casual conflation of these terms—such as treating deletion, erasure, or removal as synonymous—can lead to both technical missteps and legal non-compliance. Precise definitions matter. They determine not just how systems behave, but how agencies meet their regulatory responsibilities and avoid undermining their analytic or operational value.

The following terms are central to any discussion about personal data handling, and their distinctions form the conceptual foundation for the design recommendations presented in this guidance. They are drawn directly from definitions cited or implied in formal regulation, including the GDPR and the NZ Privacy Act, and are further detailed with their original language and sources in Appendix C. Understanding these terms with precision is critical to ensuring both technical integrity and legal compliance.

- **Deletion** is the permanent technical removal of data. It is irreversible.
- **Erasure** is a legal term under GDPR. It includes deletion or irreversible anonymisation.
- **Removal** means denying access, not necessarily deleting data.
- **Anonymisation** renders data no longer attributable to a person.
- **Pseudonymisation** masks identity with reversible tokens.

When systems fail to distinguish these terms, they conflate legality with functionality, and privacy with destruction.

Clarifying Identifiability

At the heart of privacy regulation is not data itself, but the act of identification. Regulatory obligations apply only to data that can be connected to a specific, living person—what the law defines as Personal Information. The threshold is not the presence of any one attribute, but the capacity of combined attributes to single someone out.

There are many people with blue eyes. Some share a birthday. Fewer share both. Perhaps only one person in a region carries a rare disease. This is the essence of identifiability: data becomes regulated when its combination of one or more attributes enables someone to be uniquely recognised.

In system design, the goal is not to erase all data. It is to remove the ability to identify. This misunderstanding—that deletion was required, when in fact it was identifiability that

must be addressed—led to widespread systems erasing meaning, continuity, and accountability.

However, privacy regulations, including GDPR and the NZ Privacy Act, do not require this. They require the dissociation of identity from information. This can be achieved through targeted dissociation—replacing user identifiers with a public or anonymous actor, and redacting specific fields that contribute to identifiability. The vast majority of system attributes can remain, provided they no longer link to a person.

This distinction matters. The regulation is not about whether information exists, but whether it identifies. What matters is whether the information—considered alone or in context—still allows the individual to be identified. If not, it is no longer Personal Information. The system may retain the record for accountability, analytics, or continuity, without breaching privacy obligations.

Clarifying Personal Information

It is important to clarify that regulatory obligations—under both the NZ Privacy Act and international instruments such as the GDPR—are concerned specifically with Personal Information.

Personal Information is defined as any information about an identifiable individual. If information cannot be linked to a specific, living person, it no longer falls within the regulatory definition.

This principle underpins the role of anonymisation in privacy-preserving system design. Once data is irreversibly anonymised—such that re-identification is not reasonably possible—it is no longer considered Personal Information. As such, it is no longer subject to the requirements of retention limits, access controls, or erasure.

Under the General Data Protection Regulation (GDPR), the obligation to erase personal data—outlined in Article 17—is triggered when data is no longer necessary for the purposes for which it was collected, or when consent is withdrawn and no other lawful basis exists. However, Recital 26 makes a critical clarification: data that has been rendered anonymous in such a way that the data subject is no longer identifiable is not considered personal data. As such, anonymised data falls outside the scope of the regulation. The test for anonymity is practical and contextual—it must be reasonably unlikely that re-identification can occur, taking into account current technology and the effort required.

Under NZ Privacy Principle 12, agencies are required to delete personal information once it is no longer required for lawful use. However, if data has been irreversibly anonymised or removed from operational use in a way that guarantees no link to an identifiable individual remains, it is no longer 'personal' and therefore no longer covered by the principle. This satisfies the intent of the regulation while preserving the analytic or operational value of non-personal data.

Clarifying Anonymisation

Both the NZ Privacy Act and the GDPR permit continued data retention where personal information is no longer identifiable. They do not require the destruction of information—only the removal of identifiability. Anonymisation is the architectural and legal technique by which this can be achieved.

To be compliant, anonymisation must be durable and demonstrably effective. It must not merely mask or obscure identifiers, but render them irretrievable through available means. This includes both structured identifiers (such as names, IDs, or contact details) and unstructured content (such as free-text comments that could reveal identity indirectly).

Where proper anonymisation is applied—whether through reassignment of ownership to a public or anonymous user, irreversible redaction of fields, or cryptographic separation of identifying information—the resulting data no longer constitutes personal information. This allows systems to retain their structural utility, preserve referential integrity, and support analytic or longitudinal use cases without breaching privacy obligations.

Anonymisation enables the system to forget who, without forgetting what. This distinction is at the heart of regulatory alignment and system integrity.

Resolution

To resolve these issues, we must treat information lifecycle as a set of state transitions, not destructive acts.

Records that contain Personal information can pass through stages. For example,

Draft, For Review, Rejected, Accepted, Active, Archived, Redacted, Anonymised, Replaced, and eventually Removed—where removal is not literal, but functional.

This reframing supports auditability, reversibility, and privacy, without compromising system coherence or analytical integrity.

This can be modelled using CRUST (Create, Retrieve, Update, State Transition), discarding the widely misapplied CRUD model. CRUST recognises that most information systems are not just stores but lifecycles. Workflows, approvals, audits, and publications are part of that journey. Under CRUST, deletion of private information is a special case of transition, to an anonymised state.[1]

This model preserves the value of records while ensuring they progress through purposeful and legally-aligned states.

---

[1] For the curious, refer to the UI layer BREAD acronym, noting it too could be improved by being updated to BREA/ST.

Implementation Guidance

The following practices translate the high-level design models into implementable patterns. They are particularly relevant for system architects, product owners, and vendors seeking to support legal compliance while preserving operational capability. Each addresses a specific technical or user experience concern that commonly arises when systems are required to 'forget' without compromising structure, traceability, or insight.

Preserve History through Versioning

Never overwrite or destroy records. Instead, append updated versions while retaining the full prior history (often done in a shadow history table). This supports transparency, reversibility, but also supports and builds system trust.

Make Actions Reversible

Not by deleting the latest record – but by adding a new copy of the second to last record.

This preserves auditability while reduces the need for confirmation modal dialogues, hesitation, training overhead, and user fear that limits service provision.

Treat Removal as a prior State

Extending the recommendation provided earlier, Design deletion not as destruction but as a transition through earlier well-defined inactive states.

This could include "Removed" and “Archived” —both maintaining structure and audit trails without active visibility.

Dissociate Identity, Not Data

As discussed earlier, when proceeding beyond “Removed” to “Deleted”, reassign records to an anonymous or system-level actor, redact specific fields only as necessary (most attributes can remain intact if they no longer can be used to identify a person).

Redact Free Text Responsibly

For unstructured data, use human review or AI-assisted tools to remove names and sensitive identifiers from notes and documents without deleting the record as a whole.

Procure Systems That Support Lifecycle Compliance

Require vendors and SaaS services to demonstrate data access controls, audit logging, but also, redaction capability, restoration of deletions, anonymisation. Ensure vendors understand privacy obligations and offer compliant deletion alternatives.

Additional Considerations

When applying one or more of the options recommended above, consider the following:

- Ensuring history tables entries are also anonymised.
- Train users in system-supported on-demand privacy workflows.

Legal Support

This interpretation is supported by the Irish Data Protection Commission's 2019 guidance. It clarifies that anonymisation satisfies the requirement of erasure under GDPR, provided the result is irreversible. NZ Privacy Principle 9 also supports this by stating agencies must not hold information longer than necessary—but it does not mandate deletion where anonymisation or archival serve the same purpose.

Conclusion

Deleting personal information should not be a blunt act of erasure. It should be a managed transition in a well-defined lifecycle. This document refutes the outdated belief that compliance requires or ever should be literal deletion. Instead, it offers a principled, practical, and technically feasible alternative: versioning, anonymisation, and state transitions. These approaches align with privacy legislation, protect operational integrity, and support user confidence. Government agencies must design systems that respect privacy and transparency simultaneously—by design, not by accident.

Appendices

Appendix A - Document Information

Authors & Collaborators

- Sky Sigal, Solution Architect

Versions

0.1 Initial Draft

0.2 Several changes to structure

0.3 Provided Legal Support

0.4 Added References

Images

No table of figures entries found.

Tables

No table of figures entries found.

References

Overview of the General Data Protection Regulation (GDPR)

Art. 17 GDPR – Right to erasure (‘right to be forgotten’) - General Data Protection Regulation (GDPR)

The right to erasure (Articles 17 & 19 of the GDPR) | Data Protection Commission

190614 Anonymisation and Pseudonymisation.pdf

Office of the Privacy Commissioner | Privacy Act 2020

Privacy Act 2020 No 31 (as at 30 March 2025), Public Act 22 Information privacy principles – New Zealand Legislation

There are no sources in the current document.

Review Distribution

The document was distributed for review as below:

Identity	Notes
Russell Campbell, Project Manager	

Dayton Hight, Team Manager, Digital Assurance and Cyber Security	
Alan Heward, Senior Advisor Digital Assurance	
Amy Orr, Data Enterprise Architect	
Carl Klitscher, Solution Architect	
Gareth Philpott, Solution Architect	

Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

Structure

Where possible, the document structure is guided by either ISO-* standards or best practice.

Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

Acronyms

API : Application Programming Interface.

DDD : Domain Driven Design

GDPR : General Data Protection Regulation

GUI: Graphical User Interface. A form of UI.

ICT: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

IT : acronym for Information, using Technology to automate and facilitate its management.

UI : User Interface. Contrast with API.

Terms

Refer to the project's Glossary.

Application Programming Interface : an Interface provided for other systems to invoke (as opposed to User Interfaces).

Capability : a capability is what an organisation or system must be able to achieve to meet its goals. Each capability belongs to a domain and is realised through one or more functions that, together, deliver the intended outcome within that area of concern.

Domain : a domain is a defined area of knowledge, responsibility, or activity within an organisation or system. It groups related capabilities, entities, and functions that collectively serve a common purpose. Each capability belongs to a domain, and each function operates within one.

Entity : an entity is a core object of interest within a domain, usually representing a person, place, thing, or event that holds information and can change over time, such as a Student, School, or Enrolment.

Function : a function is a specific task or operation performed by a system, process, or person. Functions work together to enable a capability to be carried out. Each function operates within a domain and supports the delivery of one or more capabilities.

Person : a physical person, who has one or more Personas. Not necessarily a system User.

Persona : a facet that a Person presents to a Group of some kind.

Quality : a quality is a measurable or observable attribute of a system or outcome that indicates how well it meets expectations. Examples include reliability, usability, and performance. Refer to the ISO-25000 SQuaRE series of standards.

User : a human user of a system via its UIs.

User Interface : a system interface intended for use by system users. Most computer system UIs are Graphics User Interfaces (GUI) or Text/Console User Interfaces (TUI).