IT Project Guidance Obligations

Version: 1.0

Purpose

This document provides a structured overview of the external obligations and expectations that apply to digital services. It serves to inform project teams, solution architects, and decision-makers of the legal, regulatory, contractual, and principled constraints that must be respected throughout the service lifecycle. The aim is to reduce the risk of avoidable non-compliance by making these constraints visible early in planning, architecture, and delivery processes.

Synopsis

Organisations operate within a tightly bounded legal and regulatory environment. This guidance document identifies and explains the obligations, agreements, commitments, standards, and principles that constrain service design and delivery. It highlights areas that require explicit consideration by service providers and delivery teams to ensure that systems are granted authority to operate while remaining lawful, trusted, and effective throughout their service life. Attention is given to obligations derived from international treaties, national law, public sector policy, and sector-specific responsibilities.

Contents

- Purpose 1
- Synopsis 1
- Contents 2

Purpose and Audience 3
Scope 3
Introduction 3
Context 4
Obligation Source and Types 4
- Legal Obligations 5
- Regulations and Standards 5
- Binding Agreements 5
- Non-Binding Commitments 6
- Sectoral and Professional Obligations 6
- Organisational Policies and Enterprise Standards 6
- Project and Contract Requirements 6
- Policies and Principles 7
  - Policies 7
  - Principles 8
- Requirements 9

Appendices 11
- Appendix A - Document Information 11
  - Versions 11
  - Images 11
  - Tables 11
  - References 11
  - Review Distribution 11
  - Audience 11
  - Structure 11
  - Diagrams 12
  - Acronyms 12
  - Terms 12

Purpose and Audience

This document is intended for all individuals involved in the specification, design, development, or approval of IT-enabled services, especially large organisations or public sector and/or government agencies. It includes those responsible for business ownership, technical design, procurement, quality assurance, change governance, and operational readiness. The document highlights external constraints that are not always visible within individual agency processes but may prevent service approval if left unaddressed.

Scope

The scope of this document includes information as to obligations, irrespective of country of delivery from or to. This includes internal systems, public-facing services, shared platforms, and outsourced components. The document references obligations and expectations derived from statute, regulation, standards, treaties, and commitments—whether binding or non-binding. It also reflects sector-specific constraints in domains such as education, health, and information management.

This guidance does not describe how to design or implement services. Rather, it provides a reference framework of constraints that shape what is considered permissible, appropriate, or mandatory.

Introduction

Before any public-facing digital service is launched, it must be evaluated to confirm that it can be responsibly operated and supported over its full intended lifespan. This evaluation requires confidence that the service has been designed with sufficient attention to operational readiness, accountability, and fitness for purpose. It must also be supported by assurance that those responsible for supporting, operating, monitoring, and maintaining the service are properly equipped and informed.

Equally important is confidence that the service complies with all relevant legal, regulatory, and ethical obligations. These may not be visible in business requirements or functional specifications but remain binding nonetheless. They include expectations relating to privacy, security, information integrity, accessibility, archiving, and transparency.

Failure to identify and meet these constraints early in the project lifecycle may result in otherwise functional systems being delayed, reworked, or denied production deployment. This is not a theoretical concern—many such systems are delayed or fail because delivery stakeholders were unaware of obligations outside their day-to-day expertise. This is not a theoretical concern—many otherwise functional systems are delayed, reworked, or abandoned because delivery stakeholders were unaware of obligations outside their day-to-day expertise. This guidance is intended to prevent that outcome by equipping decision-makers with a high-level overview of what those obligations are, why they matter, and where to find authoritative direction.

Context

very digital service is created and operated within a layered set of intersecting obligations, resembling a set of nested contexts or overlapping domains. These contexts impose constraints not only on the system itself but also on the people and processes involved in its design, delivery, and operation.

At the outermost level are international obligations. These may arise from treaties, conventions, or widely adopted norms and standards such as those issued by the United Nations or the World Trade Organisation. Although not always legally binding, they often underpin and influence national legislation.

Next are national laws. These apply not only to the system as a product but also to the people working on it. From the moment a project is initiated, all contributors—whether public servants, vendors, or contractors—must comply with the laws of the country in which they work. Later, the product itself must comply with the laws of the country in which it is developed, hosted, and ultimately consumed.

Beneath these are sector-specific constraints. These may not be laws in the narrow sense but nonetheless carry serious consequences if breached. For instance, non-compliance with financial sector rules may result in penalties or loss of ability to operate (e.g., failing to meet PCI-DSS requirements for handling card payments).

Closer to the delivery surface are internal organisational policies and standards. These are often overlooked, yet they are essential for achieving interoperability, maintaining trust, and gaining necessary internal approvals.

Additionally, obligations can be distinguished by type. Some are duties, which arise from membership in a broader system (e.g., legal compliance, public record-keeping). Others are responsibilities, which are linked to roles and granted permissions (e.g., data access, operational authority). Duties are owed; responsibilities are undertaken in exchange for specific capabilities.

This layered, role-aware context must be understood from the outset to avoid costly rework and ensure that systems are trustworthy, lawful, and supportable from day one.

Obligation Source and Types

Understanding the source of an obligation is only the beginning. In practice, obligations may be inherited, imposed, or triggered by specific design choices, tools, and deployment decisions. For example, obligations can arise:

- from the underlying components a system uses—such as open-source licences, proprietary APIs, or data services subject to foreign jurisdiction;
- from cross-border hosting and delivery arrangements that must comply with laws in each jurisdiction where the system is developed, served, or consumed;
- from timing and state changes in the system lifecycle, where different obligations apply at design, operation, archival, or decommissioning phases;
- from standards bodies, sector authorities, or professional associations that require minimum performance or assurance levels (e.g. security standards, data retention periods);
- and from the internal policies, standards, or governance models of the organisation sponsoring or approving the service.

These obligations may sometimes conflict or require interpretation. A cloud-hosted system that complies with New Zealand (NZ) privacy law may still be subject to foreign government access under foreign law. A service designed for deletion on user request may violate public recordkeeping rules. Navigating these situations requires structured review, cross-disciplinary input, and clear documentation of decisions.

It also requires recognising that not all obligations are attached to the system itself. Some are attached to roles—what a person is allowed to access or change depends on both what they are authorised to do and how well they uphold their responsibilities. Obligations must therefore be treated as dynamic, context-sensitive, and embedded in both technical and human layers of service delivery.

Legal Obligations

These are the most enforceable obligations and include legislation enacted by Parliament and regulations made under delegated authority. Legal obligations may apply internationally, nationally, or to specific domains such as education or health. They govern matters such as data privacy, recordkeeping, disclosure, and operational transparency.

Regulations and Standards

Some obligations are not primary legislation but are still legally enforceable. These include delegated regulations and formal standards that have legal backing, such as those governing information security or disability access.

Binding Agreements

These include treaties, trade agreements, and other formal international or inter-agency arrangements. Though sometimes distant from daily delivery work, these agreements may impose constraints on technology use, data sharing, or system design choices.

Non-Binding Commitments

Some expectations do not carry legal weight but are politically or culturally significant. These include adherence to international declarations (such as UNDRIP), and national commitments (such as Māori Data Sovereignty principles). These shape public expectations and ethical obligations.

Sectoral and Professional Obligations

Where government agencies operate within defined sectors, they are also subject to obligations imposed by oversight bodies, sectoral regulators, or professional standards. These may include performance thresholds, certification requirements, or data handling protocols.

Organisational Policies and Enterprise Standards

At the most immediate level, obligations also arise from an agency’s own governance arrangements, risk tolerance, and internal strategy. These may include enterprise architecture principles, patterns, and technologies, acceptable use policies, or service onboarding checklists. Even if not externally mandated, they often determine whether a service is permitted to operate.

Project and Contract Requirements

While a complete discussion of requirement types, purposes, and formats is beyond the scope of this document, a high-level overview is included here. A more detailed explanation is provided in a separate guidance document.

Requirements are contractual obligations, recommendations, permissions or prohibitions[1] on outcomes. They are defined before, during, and after procurement, and correspond to different phases of the delivery lifecycle. The IIBA's Business Analysis Body of Knowledge (BABOK) remains the industry benchmark for structuring and clarifying requirements. When followed, it helps avoid costly misunderstanding, indecision, and rework across analysis, design, development, deployment, and support.

There are several groupings of requirements, which can be categorised more or less as follows:

- **Pre-RFP**: Business (why the change and service is needed), User (how is the service be used), Quality (how is the service to made available), Capability (what capabilities must the service have), and Transitional (how will the service be delivered).

[1] MUST, SHOULD, MAY, MUST NOT

- **Pre-Procurement**: System Requirements composed of both Functional and Non-Functional Requirements, defining what the system must do and how well.
- **Post-Procurement**: Technical requirements addressing specific technologies, hosting, integration, or operational dependencies.
- **During Development**: Real-world changes, missed considerations, or evolving understanding often result in new work. These take the form of **Work Items**—scoped, prioritised tasks or fixes that must be defined, scheduled, and delivered during the project. Agile methodologies are typically used to manage this emergent workload and absorb late-breaking insights or adjustments without losing project momentum.

An all too common error is not knowing about and therefore the distinction between Non-Functional System Requirements (NFRs) and Transitional Requirements. NFRs are Quality requirements that specify acceptable performance, usability, integrity, and reliability standards of the *service*. Transitional requirements, by contrast, govern the *project*—not the service or its underlying system. Transitional requirements often include data migration, access provisioning, media preparation, temporary licences or rentals, communications, and change management – all aspects that cannot be addressed by the system itself or its provider.

Another common – often critical -- error is incorrectly understanding Agile as being a substitute for early planning, using it to jump directly into delivery. This bypasses foundational scoping, sequencing, and clarity, leading to underestimations that in turn produce Missing Valuable Planning (MVP)s.

A mature organisation reuses standardised requirements wherever possible—especially for qualities—and adds project-specific refinements only when necessary to address project specific unique conditions or elevated expectations. ISO 25010, 25012, and 25022 provide useful models for consistent definition and evaluation of Non-Functional Requirements. To proceed without reliance on a these standards is to unnecessarily implement incompleteness of thought or planning.

Policies and Principles

Policies and principles are both forms of obligation, but they serve different purposes and respond to different conditions.

Policies

A policy encodes a route through *charted* territory. It is a decision that has already been made based on known conditions, risk appetite, and organisational precedent. Policies define what must be done, how it is to be done, and often in what sequence. Policies are by their very nature fixed.

Policy typically vary organisation and are not treated in depth here.

While policies play an important role in enforcing consistency and compliance, they tend to be rigid, require oversight, and can slow responsiveness. Principles, by contrast, are fewer in number, portable across projects, and allow disparate teams to act with aligned intent without requiring constant coordination or external enforcement. They enable faster, more accountable decision-making under changing or ambiguous conditions, making them especially valuable in complex or evolving delivery environments.

Principles

Principles, unlike policies, are not fixed instructions. They are structured, pre-agreed decision-making frameworks—obligations of thought—that support consistent reasoning when navigating uncertainty or novelty. They exist to guide interpretation and resolve ambiguity, particularly when operating in unfamiliar or evolving circumstances.

Where policies encode settled routes through charted territory, principles offer insight into how to chart new routes when none yet exist. A good principle makes visible the underlying rationale behind decisions, allowing diverse teams to act consistently even when circumstances differ.

For example, a principle like “Services are designed for consumers before operators” does not dictate an exact sequence of tasks but instead clarifies a consistent prioritisation logic. It makes the rationale explicit so that trade-offs and options can be assessed in alignment with shared priorities.

Principles reflect values and priorities. They may be defined at enterprise, programme, or project level and require active application—thinking, not merely following. They are not rules, but structured heuristics, intended to be enduring across project stages, architectures, and personnel turnover.

A common mistake is to treat all principles as one large list. In practice, principles should be few in number (ideally no more than six per domain) and categorised by area of concern. Domains might include Evaluation, Funding, Security, Privacy, Accessibility, Development, Integration, Onboarding, and Operation. Stakeholders should only need to internalise those relevant to their role. For instance, a developer may apply Development and Security principles, while an operator may follow Privacy and Operational principles.

Principles should not be confused with policy, standards, or values. They enable scalable decision-making in novel contexts. A principle like “prefer reversibility over commitment” does not dictate a design but shapes how trade-offs are framed. A different context—say, an emergency system—may prioritise stability instead.

Principles are a discipline. They demand clarity of thought, consistency of application, and deliberate maintenance. When well-formed, they reduce delay, disagreement, and incoherence during delivery, especially where conditions are complex or change is rapid.

Requirements

While a complete discussion of requirement types, purposes, and formats is beyond the scope of this document, a high-level overview is included here. A more detailed explanation is provided in a separate guidance document.

Requirements are contractual obligations on outcomes. They are defined before, during, and after procurement, and correspond to different phases of the delivery lifecycle. The IIBA's Business Analysis Body of Knowledge (BABOK) remains the industry benchmark for structuring and clarifying requirements. When followed, it helps avoid costly misunderstanding, indecision, and rework across analysis, design, development, deployment, and support. At a minimum, requirements should distinguish between Business, User, System, and Transitional needs—an approach often abbreviated as BUST.

There are several phases in which requirements are developed, which can be grouped more or less as follows:

- **Pre-RFP**: *Business* (why the change and service is needed), *User* (how is the service be used), *Qualities* (how is the service to made available), *Capabilities* (what capabilities must the service have), and [Delivery] *Transitional* (how will the service be delivered).
- **Pre-Procurement**: *System* Requirements, composed of both *Functional* and *Non-Functional* Requirements, defining what the system must do and how well.
- **Post-Procurement**: *Technical* requirements addressing specific standards, technologies, hosting, and integration dependencies.
- **During Development**: Real-world changes, missed considerations, or evolving understanding often result in new work. These take the form of *Work Items*—scoped, prioritised tasks or fixes that must be defined, scheduled, and delivered during the project. Agile methodologies are typically used to manage this emergent workload and absorb late-breaking insights or adjustments without losing project momentum.

A common almost industry wide error is treating Agile as a substitute for early planning, using it to jump directly into delivery. This bypasses foundational scoping, sequencing, and clarity—resulting in Missing Valuable Planning (MVP).

A common error is confusing Non-Functional System Requirements (NFRs) with Transitional ones. NFRs are Quality requirements that specify acceptable performance, usability, integrity, and reliability standards. Transitional requirements, by contrast, govern the project—not the system. These include data migration, access provisioning, media preparation, temporary licences or rentals, communications, and change management.

A mature organisation reuses standardised requirements wherever possible—especially for qualities—and adds project-specific refinements only when necessary to address project specific unique conditions or elevated expectations. ISO 25010, 25012, and 25022 provide useful models for consistent definition and evaluation of Non-Functional Requirements.

Appendices

Appendix A - Document Information

Authors & Collaborators

- Sky Sigal, Solution Architect

Versions

0.1 Initial Draft

0.2 1.0

Images

No table of figures entries found.

Tables

No table of figures entries found.

References

There are no sources in the current document.

Review Distribution

The document was distributed for review as below:

Identity	Notes
Russell Campbell, Project Manager	
Amy Orr, Solution Architect	
Carrie Buckmaster, Senior Business Analyst	
Alan Heward, Senior Compliance Advisor	

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

GUI: Graphical User Interface. A form of UI.

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.

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.

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).