Contents

Project Management

1. Why Project Management

Even a well-designed system will fail if the project is poorly managed.

So Project Management is treated as:

- A **governance control**, and
- A **risk management tool** for ICT projects.

It applies **across the entire system lifecycle**, from planning to disposal.

2. What Project Management is about

"Project management is about planning, coordinating, and controlling ICT work so systems are delivered on time, within budget, and with the expected quality."

In Government ICT, it ensures:

- Clear accountability,
- Controlled spending,
- Predictable outcomes,
- Reduced failure rates.

3. What institutions are expected to do

1. Assign clear project ownership and leadership

("Who is responsible?")

Institutions must:

- Appoint a **Project Sponsor** (usually from business),
- Assign a **Project Manager**,
- Clearly define roles and responsibilities.

No project should run without accountable leadership.

2 Plan the project before execution

("Know what you're doing before you start")

Institutions are expected to:

- Define project scope,
- Set timelines and milestones,
- Estimate costs and resources,
- Identify risks.

This ties directly to **Planning, Analysis and Design**.

3 **Monitor progress and control changes**

(“Stay in control”)

Institutions must:

- Track progress against plans,
- Manage scope changes formally,
- Address risks and issues promptly.

This prevents:

- Scope creep,
- Cost overruns,
- Delayed delivery.

4 Ensure alignment with development tracks

(“Manage projects according to risk”)

Project management rigor must:

- Match the **Systems Development Track** (6.3.1.22),
- Be stricter for high-risk systems,
- Be lighter for simple systems.

Project management is **proportional**, not one-size-fits-all.

5 Coordinate stakeholders and users

(“Keep everyone involved”)

Institutions must:

- Involve users and functional offices,
- Coordinate vendors and ICT teams,
- Manage expectations throughout the project.

This improves acceptance and reduces resistance.

6 Close the project formally

("Finish properly")

At the end of development or deployment, institutions must:

- Confirm deliverables are complete,
- Obtain acceptance and sign-off,
- Document lessons learned.

Projects should not "fade out" informally.

4. What ICTA is really trying to prevent

By including Project Management, ICTA is addressing common failures such as:

- ICT projects with no clear owner,
- Endless development with no milestones,
- Cost overruns with no accountability,
- Systems delivered but never formally accepted.

5. What auditors usually check here

Typical audit questions include:

- Was a project sponsor and manager appointed?
- Is there a project plan and budget?
- Were risks and changes managed?
- Was the project formally closed?

Planning, Analysis and Design Process

1. What this phase is about

"Before Government builds or buys any system, it must fully understand the problem, define what the system should do, and agree on how the solution will look — both from a business and technical perspective."

This phase exists to ensure that:

- Everyone understands the same problem,
- The solution is clearly thought through,
- There is agreement **before money is spent or code is written**.

2. How the Standard structures this phase (Annex 6)

Annex 6 breaks this phase into **two critical sub-phases** that together form Planning, Analysis and Design:

1. **Requirements Definition Phase**
2. **System Definition Phase**

These two phases are the **official backbone** of planning and design in the Standard.

3. Requirements Definition Phase (Planning & Analysis)

This phase is about **understanding the business need** and agreeing on **what the system must achieve**, without jumping into technical solutions too early.

In simple terms:

"This is where Government clearly defines what it wants and why it wants it."

What institutions are expected to do

According to Annex 6, institutions must document the following:

A. Scope and objectives

Institutions must clearly state:

- The overall purpose of the system,
- What the system is intended to achieve,
- Any changes from earlier proposals.

This ensures the system has a **clear justification**.

B. Identify users and stakeholders

The institution must:

- Identify all functional offices that will use the system,
- Describe how each office will use it (e.g. read-only, data entry, approvals).

This prevents systems that ignore real users.

C. Describe the current system or procedures

Institutions must document:

- Existing systems or manual processes,
- Weaknesses or gaps in the current setup,
- Differences between current and proposed processes.

This ensures the system solves **real problems**, not imagined ones.

D. Define data sources

Institutions must clearly describe:

- Where data will come from,
- Who owns each data source,
- How data accuracy and integrity will be ensured.

This directly links to the **Information Viewpoint**.

E. Describe processing and outputs

The institution must explain:

- What processing the system will perform,
- How data will be updated and managed,
- What outputs will be produced (reports, interfaces, inquiries).

This forms the basis of later system functionality.

Output of this phase

By the end of Requirements Definition, there should be a **Requirements Definition Document** that both business and ICT teams understand and agree on.

4. System Definition Phase (Design)

What this phase is about

This phase moves from *"what do we need?"* to *"how will the system work?"* — but still at a **high-level**, not detailed coding.

In simple terms:

"This is where the system is designed enough for everyone to visualize it and agree before development or procurement begins."

What institutions are expected to do

Annex 6 requires the following to be defined:

A. System architecture

Institutions must describe:

- The type of system (web-based, client/server, standalone, etc.),
- How users will access the system,
- How components relate to each other.

This links to the **Computational and Engineering Viewpoints**.

B. Functional and technical overview

The system definition must:

- Describe key system functions,
- Show how business needs will be met,
- Explain technical approaches not obvious from code.

This ensures shared understanding between ICT and business teams.

C. Data structure and databases

Institutions must describe:

- Databases involved,
- Major data entities and relationships,
- Data elements at a high level.

This ensures data is not an afterthought.

D. Agreement before prototyping or development

Annex 6 emphasizes that:

- The System Definition must be agreed upon by both functional offices and developers,
- Prototyping or development may proceed iteratively, but **only after this shared understanding exists**.

This prevents disputes later in the project.

Output of this phase

The key output is a **System Definition Document**, which:

- Bridges business and technical understanding,
- Guides acquisition or development,

- Forms the baseline for later phases.

5. What the Standard is very clear about

Two important clarifications from Annex 6:

1. **Do not decide technology too early**
   The Standard explicitly warns against locking into vendors or technologies before requirements are clear.
2. **Methodology flexibility is allowed**
   Institutions may use Agile, Waterfall, or other models — as long as these **planning and design outcomes exist**.

6. In Summary

"The Planning, Analysis and Design process requires institutions to first document what they need through a Requirements Definition, and then agree on how the system will work through a System Definition. These steps are detailed in Annex 6 and must be completed before any system is built or acquired."

7. Why this phase is critical from an audit perspective

This is the **foundation phase** ICTA looks for. Typical audit evidence includes:

- Requirements Definition Document,
- System Definition Document,
- Evidence of stakeholder involvement,
- Evidence of approval before acquisition or development.

If these are missing, it usually indicates **non-compliance at the very start of the system lifecycle**.

Application and System Software Acquisition

1. What this section is about

"This section explains how Government can obtain systems — whether by building them, buying them, or outsourcing them — as long as the process is controlled, justified, and compliant."

The key point is this:

Government is not forced into one acquisition method.
What matters is that **the method chosen is appropriate, justified, and properly governed**.

2. Why the standard allows different acquisition tracks

The standard recognises that:

- Not all institutions have the same ICT capacity,
- Not all systems carry the same risk,
- Not all solutions need to be built from scratch.

So instead of prescribing a single approach, it allows **multiple acquisition tracks**, as long as:

- Planning and design were done properly
- The acquisition aligns with architecture and standards,
- Risks and costs are understood.

3. The main acquisition tracks recognised by the standard

Track 1: In-house development

(Build it internally)

What this means

The institution designs, develops, tests, and maintains the system using its **own ICT staff**.

When this track makes sense

- The institution has strong internal ICT capacity,(In Present terms, this may refer to the use of ICT Capacity within Gvernment)

- The system is highly specific to the institution's mandate,
- There is a need for flexibility and frequent changes.

What the standard expects

Even though it's internal, the institution must still:

- Follow the planning and design steps,
- Document requirements and system definition,
- Apply testing, change management, and maintenance controls.

Track 2: Outsourced / Contracted development

(Build it through a vendor)

What this means

A vendor is contracted to design and develop the system based on approved requirements.

When this track makes sense

- The institution lacks sufficient internal development capacity,
- The system is complex or time-sensitive,
- External expertise is required.

What the standard expects

The institution must:

- Retain ownership of requirements and design decisions,
- Ensure contracts clearly define deliverables,
- Avoid vendors driving architecture without oversight.

The system must still comply with ICTA standards, **even if a vendor builds it**.

Track 3: Commercial Off-The-Shelf (COTS) software

(Buy an existing system-Commercial)

What this means

The institution acquires an already-built solution and configures it to meet its needs.

When this track makes sense

- The business problem is common across Government,
- Proven solutions already exist,
- Custom development would be wasteful.

What the standard expects

Institutions must:

- Ensure the software meets requirements,
- Assess licensing, support, and upgrade implications,
- Avoid excessive customization that breaks maintainability.

Buying software does **not** remove the need for planning and analysis.

Track 4: Configuration / Customisation of existing Government platforms

(Reuse before building)

What this means

Instead of acquiring a new system, the institution:

- Extends,
- Integrates with, or
- Configures an existing Government system or shared service.

When this track makes sense

- A similar system already exists within Government,
- Integration is cheaper and faster than building new,
- Whole-of-government consistency is desired.

What the standard expects

Institutions must:

- Assess compatibility and integration impact,

- Ensure data ownership and security are clear,
- Avoid duplicating functionality already available.

This track directly supports **shared services and interoperability**.

Track 5: Leased / Subscription-based solutions (e.g. SaaS)

(Use but don't own the software)

What this means

The institution uses software hosted and managed by a third party, usually under a subscription model.

When this track makes sense

- Rapid deployment is needed,
- Infrastructure capacity is limited,
- Costs need to be predictable.

What the standard expects

Institutions must:

- Address data ownership and sovereignty,
- Ensure compliance with security and privacy requirements,
- Have exit and migration strategies.

Convenience does **not** override governance.

4. What the standard is very firm about (across all tracks)

Regardless of the acquisition track chosen, the standard insists that:

- Planning, analysis, and design **must come first**,
- Architecture and interoperability must be respected,
- Legal, licensing, and ownership issues must be clear,
- Maintenance and disposal must be planned from the start.

Application and System Software Development

1. What this section is about (plain explanation)

At this stage, Government has:

- Completed planning, analysis, and design,
- Chosen an acquisition approach,
- Approved requirements and system definition.

This section now focuses on **how the system is actually built or configured**, but in a **controlled, risk-aware way**.

The standard recognises that:

Not all systems are equal, so development controls must match the system's size, complexity, and risk.

2. Systems Development Tracks

Before development starts, the institution must **deliberately choose a development track** based on risk and complexity.

The three tracks recognised under 6.3.1.22

1. **Simple Systems Development Track**
    - Small, low-risk, non-critical systems
    - Minimal impact if the system fails
2. **Moderately Complex Systems Development Track**
    - Core operational systems
    - Moderate risk and integration needs
3. **Large, Complex or High-Risk Systems Development Track**
    - Mission-critical, high-impact systems
    - Failure would severely affect Government operations

The **chosen track determines how strict development controls must be**.

3. The Six Key Things Institutions Must Do During Development

1️⃣ Develop strictly against approved requirements

(Build what was agreed)

Institutions must ensure:

- Development follows the approved **Requirements Definition** and **System Definition**,
- Any changes go through formal **change control**.

2️⃣ Follow an appropriate development methodology

(Use a structured approach)

The standard does **not mandate a specific SDLC**, but requires that:

- The chosen methodology is documented,
- Outputs (designs, builds, tests) are traceable.

3️⃣ Separate development, testing, and production environments

(Protect live systems)

Institutions must:

- Keep development, testing, and production environments separate,
- Prevent untested code from reaching live systems.

4️⃣ Build security into development

(Secure by design)

Security must be considered throughout development, including:

- Access controls,
- Secure coding practices,
- Protection of credentials and data.

5️⃣ Document the system properly

(Make it understandable and maintainable)

Institutions must produce:

- Technical documentation,
- User guides,
- Configuration and support documentation.

6️⃣ Involve users during development

(Build with users, not in isolation)

Institutions are expected to:

- Involve users in reviews and validation,
- Confirm the system meets real operational needs.

4. In Summary

"Once Government starts building a system, it must follow a structured development process. The standard recognises three development tracks—simple, moderate, and high-risk—and requires institutions to apply controls that match the system's importance. Regardless of the track, systems must be built against approved requirements, securely, with proper testing, documentation, and user involvement."

5. Why this matters from an audit and governance perspective

This section allows auditors to ask very clear questions:

- Was a **development track chosen and justified**?
- Did development controls match the **risk level of the system**?
- Were requirements, security, testing, and documentation properly handled?

Application and System Software Testing, Acceptance and Deployment

1. What this section is about (plain explanation)

"This part is about making sure a system actually works as intended, is accepted by users, and is safely released into live use."

The standard treats testing and deployment as:

- A **quality gate**, and
- A **risk control**.

No system should go live simply because development is "complete".

2. Why ICTA places strong emphasis here

Many system failures happen because:

- Systems are rushed into production,
- Users never validated them,
- Critical scenarios were never tested.

So this section ensures:

- Errors are detected early,
- Users formally accept the system,
- Deployment does not disrupt services.

3. The three components explained

A. Testing

("Does the system work?")

Testing ensures the system:

- Meets functional requirements,
- Performs reliably,
- Is secure and stable.

What institutions are expected to do

Institutions must:

- Prepare a testing plan,
- Test against approved requirements,
- Document test results and defects.

Types of testing expected (depending on system track)

- Functional testing
- Integration testing
- Performance testing
- Security testing

B. User Acceptance

(“Are users satisfied?”)

User Acceptance Testing (UAT) ensures:

- The system supports real work,
- Business users confirm readiness,
- There is formal approval to go live.

What institutions are expected to do

Institutions must:

- Involve actual users,
- Document acceptance criteria,
- Obtain formal sign-off.

UAT is a **business decision**, not a technical one.

C. Deployment

(“How does it go live?”)

Deployment is about:

- Moving the system into production safely,
- Minimising disruption,
- Ensuring support readiness.

What institutions are expected to do

Institutions must:

- Have a deployment plan,
- Backup data before go-live,
- Ensure rollback procedures exist.

4. How testing and deployment vary by development track

Aspect	**Simple Track**	**Moderate Track**	**High-Risk Track**
Test planning	Basic	Formal	Detailed
Functional testing	Required	Required	Extensive
Integration testing	Limited	Required	Extensive
Security testing	Basic	Required	Mandatory
UAT	Informal	Formal sign-off	Structured sign-off
Deployment controls	Simple	Controlled	Strict & staged

5. What institutions must NOT do

The standard is very clear on what should never happen:

- No testing directly in production,
- No go-live without user acceptance,
- No undocumented deployments.

6. In Summary

"Before a system goes live, it must be tested against requirements, accepted by users, and deployed in a controlled way. The depth of testing depends on how risky the system is, but no system should go live without proper testing and approval."

7. Why this section is critical from an audit perspective

Auditors typically look for:

- Test plans and test reports,
- Evidence of UAT and sign-off,
- Deployment plans and approvals.

A common audit finding is:

Systems going live with no documented testing or user acceptance.

Application and System Software Maintenance

1. What this section is about

"This part is about how Government looks after a system after it goes live, so it continues working, stays secure, and adapts to change."

The Standard recognises a simple truth:

Most system problems happen after go-live, not during development.

Maintenance should be treated as a **formal, ongoing responsibility**, not an afterthought.

2. Why ICTA places strong emphasis on maintenance

ICTA has seen many systems fail because:

- They are deployed and then neglected,
- Changes are done informally,
- Vendors disappear with system knowledge,
- Security updates are ignored.

This section exists to ensure:

- Systems remain reliable and secure,
- Changes do not introduce new risks,
- Government retains control over its systems.

3. What institutions are expected to do during maintenance

(The six core maintenance obligations)

1️⃣ Establish clear maintenance ownership and support arrangements

("Who is responsible for the system?")

Institutions must:

- Clearly assign responsibility for system maintenance,
- Decide whether maintenance is:
    - Internal,
    - Vendor-supported, or
    - Hybrid,
- Ensure continuity of support.

In practice, this means:

- Maintenance contracts or SLAs,
- Defined ICT and business roles,
- Budget allocation for maintenance.

A system with no clear owner is non-compliant by default.

2️⃣ Control and document all system changes

("No silent fixes")

All maintenance activities must be:

- Requested formally,
- Reviewed for impact,
- Approved before implementation.

This applies to:

- Bug fixes,
- Enhancements,
- Configuration changes,
- Performance tuning.

In practice, this means:

- Change request forms,
- Impact assessments,

- Approval records.

This prevents uncontrolled changes that break systems.

3️⃣ Test maintenance changes before applying them to production

("Fix safely")

Institutions must:

- Apply fixes in a test or staging environment first,
- Validate that changes do not break existing functionality,
- Only then deploy to production.

In practice, this means:

- Separate test/maintenance environments,
- Rollback procedures,
- Deployment approvals.

Direct fixes in production are a **major red flag**.

4️⃣ Apply security patches and updates promptly

("Stay secure over time")

Maintenance must explicitly include:

- Operating system updates,
- Application patches,
- Library and framework updates,
- Vulnerability remediation.

The Standard treats security maintenance as **mandatory**, not optional.

In practice, this means:

- Patch schedules,
- Security monitoring,

- Incident response readiness.

5️⃣ Keep system documentation up to date

("Documentation must evolve with the system")

As systems change, institutions must update:

- Technical documentation,
- User manuals,
- Configuration and operational guides.

This ensures:

- Knowledge is institutionalised,
- Systems can be supported even if staff or vendors change.

Outdated documentation is treated as poor maintenance.

6️⃣ Monitor system performance and usage

("Is the system still fit for purpose?")

Institutions must:

- Monitor uptime and performance,
- Track system errors and failures,
- Identify when enhancements or upgrades are needed.

This helps:

- Detect emerging risks early,
- Justify future improvements or replacement.

4. Maintenance and system risk (important linkage)

Just like development:

- **Maintenance controls must match system risk and complexity**

System Type	Maintenance Rigor
Simple systems	Basic controls
Core operational systems	Formal maintenance procedures
High-risk / mission-critical systems	Strict controls, SLAs, audits

High-risk systems require **stronger monitoring, security, and change controls**.

5. What the Standard is very clear about

The Standard implicitly prohibits:

- Informal system changes,
- Unapproved fixes,
- Unsupported systems running in production.

6. What auditors usually check here

Typical audit evidence includes:

- Maintenance contracts or SLAs,
- Change logs and approvals,
- Patch and update records,
- Updated documentation,
- System monitoring reports.

A common audit finding is:

Systems running for years with no formal maintenance controls.

Application and System Software Retirement

1. What this section is about

"This part is about how Government safely stops using a system when it is no longer needed, outdated, or being replaced."

In simple terms:

- Systems should **not live forever**,
- Keeping obsolete systems running creates:
    - Security risks,
    - High maintenance costs,
    - Data integrity issues.

So retirement is about **ending a system's life in a controlled, safe, and documented way**.

2. Why ICTA emphasizes system retirement

The standard recognises common Government problems such as:

- Old systems still running "just in case",
- Unsupported software exposed to cyber threats,
- Data scattered across legacy systems,
- No clear decision on when systems should be switched off.

This section exists to ensure that:

- Systems are retired deliberately,
- Data is preserved or disposed of lawfully,
- Risks are reduced, not increased.

3. When a system should be retired

Institutions are expected to consider retirement when:

- The system is obsolete or no longer supported,

- A new system has replaced it,
- Maintenance costs outweigh benefits,
- The system no longer aligns with institutional mandate,
- Security risks can no longer be reasonably managed.

Retirement is therefore a **business and risk decision**, not just a technical one.

4. What institutions are expected to do during retirement

(The six core retirement controls)

1️⃣ Make a formal decision to retire the system

("This system will be switched off")

Institutions must:

- Formally approve the retirement decision,
- Clearly document the reasons for retirement,
- Identify timelines and responsibilities.

Systems should never be abandoned informally.

2️⃣ Plan data migration, archiving, or disposal

("What happens to the data?")

Institutions must decide:

- Which data must be migrated to a new system,
- Which data must be archived for legal or audit purposes,
- Which data can be securely destroyed.

This must align with:

- Records management laws,
- Data protection requirements.

Data handling is the **most critical part** of retirement.

3️⃣ Ensure continuity of service

(“No service disruption”)

Before switching off a system, institutions must:

- Ensure replacement systems are operational,
- Confirm users can continue their work,
- Avoid service gaps.

This is especially important for citizen-facing systems.

4️⃣ Securely decommission the system

(“Shut it down safely”)

Institutions must:

- Disable system access,
- Decommission servers and infrastructure,
- Remove credentials and integrations.

This prevents:

- Unauthorized access,
- Data leaks,
- Shadow systems.

5️⃣ Update documentation and system registers

(“Close the record”)

Institutions must:

- Update ICT asset registers,
- Record the system’s retirement status,

- Retain relevant documentation for audit purposes.

Retired systems should be clearly marked as such.

6️⃣ Terminate licenses and support contracts

("Stop paying for what you no longer use")

Institutions must:

- Cancel software licenses,
- Terminate vendor support contracts,
- Ensure no unnecessary costs continue after retirement.

This supports value for money and good governance.

5. What the Standard is very clear about

The Standard implicitly warns against:

- Leaving systems running "just in case",
- Retiring systems without handling data properly,
- Losing audit trails and historical records.

6. Why this section matters from an audit perspective

Auditors typically look for:

- Formal retirement approvals,
- Data migration or archiving evidence,
- Updated asset registers,
- Terminated licenses and contracts.

A common audit finding is:

Legacy systems still running with no owner, no support, and high security risk.

Application and System Software Disposal

1. What "Disposal" means (layman explanation)

"Disposal is about permanently getting rid of a system and its associated software, data, licenses, and infrastructure in a secure and lawful way."

In simple terms:

- **Retirement** = *Stop using the system*
- **Disposal** = *Safely and permanently remove it*

A system can be **retired but not yet disposed** (for example, data is still archived), but disposal is the **final irreversible step**.

2. Why ICTA separates Disposal from Retirement

ICTA separates these two because:

- Retirement focuses on **business use**,
- Disposal focuses on **risk, security, and compliance**.

Improper disposal can lead to:

- Data leaks,
- Legal violations,
- Unauthorized reuse of Government systems,
- Continued licensing or cloud costs.

So disposal is treated as a **high-risk activity that must be tightly controlled**.

3. When Disposal should happen

Disposal should occur when:

- A system has been formally retired,
- All required data has been migrated or archived,

- Legal and audit retention requirements have been met,
- There is no further operational or legal need for the system.

Disposal should **never happen before retirement**, but it must eventually follow it.

4. What institutions are expected to do during Disposal

(The six core disposal controls)

1️⃣ Obtain formal approval to dispose of the system

(“We are permanently removing this system”)

Institutions must:

- Formally approve the disposal decision,
- Confirm that retirement conditions have been met,
- Document authorization and accountability.

Disposal without approval is a serious governance failure.

2️⃣ Securely destroy or sanitize data and storage media

(“Ensure data cannot be recovered”)

Institutions must:

- Securely erase data from storage devices,
- Apply approved data destruction or sanitization methods,
- Ensure no residual data remains.

This applies to:

- Databases,
- Backups,
- Portable media,
- Cloud storage.

This is the **most critical disposal requirement**.

3️⃣ Decommission and dispose of software components

(“Remove the software completely”)

Institutions must:

- Uninstall applications and middleware,
- Remove system configurations,
- Disable integrations and APIs.

This ensures:

- The system cannot be reactivated,
- No hidden access paths remain.

4️⃣ Terminate licenses, subscriptions, and access rights

(“Close all doors and stop all costs”)

Institutions must:

- Terminate software licenses,
- Cancel SaaS subscriptions,
- Revoke user and system access credentials.

This prevents:

- Ongoing costs,
- Unauthorized access,
- Vendor dependency after disposal.

5️⃣ Dispose of or repurpose infrastructure securely

(“Handle hardware and cloud resources properly”)

Institutions must:

- Decommission servers (on-prem or cloud),
- Securely wipe or destroy hardware if applicable,
- Ensure repurposed assets do not contain residual data.

This includes:

- Servers,
- Storage devices,
- Virtual machines.

6️⃣ Update records and retain disposal evidence

("Close the lifecycle properly")

Institutions must:

- Update ICT asset registers,
- Record disposal details,
- Retain disposal certificates or logs for audit purposes.

If it's not documented, it's assumed not to have happened.

5. Key difference: Retirement vs Disposal (simple table)

Aspect	**Retirement**	**Disposal**
Purpose	Stop using the system	Permanently remove it
Data	Still retained or archived	Destroyed or sanitized
Reversibility	Sometimes reversible	Irreversible
Risk focus	Service continuity	Security & compliance
Audit risk	Medium	High

6. Why Disposal is a high-risk audit area

Auditors typically check:

- Approval for disposal,
- Evidence of data destruction,
- License termination records,
- Updated asset registers.