Architecture Description Standard (ADS)

Version 1.3.2

Andi Chandler

2026-06-30

Contents

Architecture Description Standard

Version 1.3.2 | **Author:** Andi Chandler | **Licence:** CC BY 4.0

Published by: ArchStandard (archstandard.org)

Overview

The Architecture Description Standard (ADS) defines how to write a Solution Architecture Document (SAD) — the single document that describes how a technology solution is designed, built, and operated.

Purpose

This standard defines the required structure and content for **Solution Architecture Documents (SADs)**. The architectural views and quality attributes within a SAD constitute the **High Level Design (HLD)** of the solution.

A Solution Architecture Document conforming to this standard serves the following functions:

1. **Communication** — Provides a shared understanding of the solution architecture across all stakeholders
2. **Governance** — Enables the architecture governance process to evaluate the design against quality criteria
3. **Traceability** — Establishes links between design decisions, business requirements, quality attributes, and applicable standards
4. **Reference** — Acts as a living document that accurately describes the current-state architecture
5. **Evidence** — Demonstrates that the solution is well-architected against industry-recognised frameworks

Scope

This standard applies to the documentation of any technology solution architecture, regardless of:

- **Deployment model** — cloud-native, on-premises, hybrid, multi-cloud, or SaaS
- **Scale** — from single applications to multi-application ecosystems
- **Industry** — the standard is organisation-agnostic and sector-neutral
- **Lifecycle stage** — new builds, migrations, modernisations, or incremental changes

The standard is **extensible** — organisations can add custom sections, map to internal tools and standards, and define governance gates without modifying the core structure. See Organisation Customisation for details.

Audience

This standard is intended for use by:

Role	Primary Sections
Solution Architects	All sections — primary authors of SADs
Enterprise Architects	Framework Alignment, Quality Attributes, Governance
Security Architects	Section 3.5 (Security View), Quality Attributes
Infrastructure Engineers	Section 3.3 (Physical View), Quality Attributes
Development Leads	Section 3.1 (Logical View), Section 5 (Lifecycle)
Architecture Governance	All sections — as a review and governance framework (e.g., ARB, design authority)

Document Structure

A Solution Architecture Document conforming to ADS is organised into three tiers:

Tier 1 — Architectural Views (Sections 3.1 – 3.6)

The architecture is described through complementary views, each addressing different stakeholder perspectives. These are based on Kruchten's 4+1 Architectural View Model — a widely-used framework that organises architecture documentation into Logical, Process, Development, and Physical views, plus Scenarios that tie them together — extended here with dedicated Data and Security views.

Together, these views form the **High Level Design (HLD)** of the solution:

View	Viewpoint	Primary Stakeholders
3.1 Logical View	Application architecture, components, patterns	Architects, Developers
3.2 Integration & Data Flow View	Data flows, integrations, interfaces	Integrators, Architects
3.3 Physical View	Deployment, hosting, networking, environments	Infrastructure, DevOps
3.4 Data View	Data stores, classification, privacy, lifecycle	Data Architects, Compliance
3.5 Security View	IAM, encryption, monitoring, threat model	Security, CISO, Compliance
3.6 Scenarios	Key use cases, architecture decision records	All Stakeholders

No single view provides a complete description of the architecture. Together, the views provide a holistic picture addressable by all stakeholder concerns.

Tier 2 — Quality Attributes (cross-cutting)

Evaluate the cross-cutting quality attributes **across** all architectural views. These are derived from the cloud Well-Architected Frameworks:

Quality Attribute	Focus
4.1 Operational Excellence	Observability, monitoring, operational procedures
4.2 Reliability & Resilience	DR, scalability, fault tolerance, backup and recovery
4.3 Performance Efficiency	Performance requirements, resource optimisation
4.4 Cost Optimisation	Cost analysis, FinOps practices
4.5 Sustainability	Energy efficiency, carbon impact, resource efficiency

Tier 3 — Lifecycle & Governance

Lifecycle Management (Section 5)

Describes how the solution is developed, deployed, operated, and eventually retired — including CI/CD, migration strategy, resourcing, release management, and exit planning.

Decision Making & Governance (Section 6)

Captures the constraints, assumptions, risks, dependencies, and issues that shaped the design. Documents key architecture decisions (ADRs), guardrail exceptions, and compliance traceability.

How the Sections Relate

Reading order: A completed SAD tells a story. It begins with the business context and who cares about the solution (Sections 0-2). It then describes the architecture itself through multiple views (Section 3). It evaluates how well the architecture performs against quality attributes (Section 4). It explains how the solution is built, deployed, and operated (Section 5). Finally, it records the decisions, risks, and governance that shaped the design (Section 6).

The diagram below shows how the three tiers work together. Architectural views describe *what* the solution is. Quality attributes assess *how well* it performs. Lifecycle and governance cover *how* it is built, run, and governed.

Documentation Depths

This standard assigns each section a documentation depth — **[Minimum]** (SHALL), **[Recommended]** (SHOULD), or **[Comprehensive]** (MAY) — that defines when it must be completed. This enables progressive adoption: begin with Minimum for early-stage designs, expand to Comprehensive for critical and regulated systems.

See Conformance and Usage for the full depth table, terminology definitions, and governance gate mapping.

Conformance

A Solution Architecture Document **conforms** to this standard when:

1. All sections marked **[Minimum]** are completed
2. The document structure follows the section numbering and hierarchy defined herein

3. Architectural views address the stakeholder concerns identified in Section 2
4. Quality attributes are evaluated as cross-cutting lenses across the architectural views
5. The document is validated against the JSON Schema (ADS Schema v1.0.0) if machine-readable output is required

A document that additionally completes all **[Recommended]** sections achieves **full conformance**.

Ready to Start?

- **Quickstart** — create your first SAD in minutes
- **Depth Cheat Sheet** — see exactly what to fill in at each depth
- **Download a template** — Word, Markdown, YAML, or JSON
- **See completed examples** — four worked examples across different project types

Contributing

ADS is open-source and hosted on GitHub. Contributions are welcome — submit issues, suggest improvements, or open a pull request. The standard content is licensed under CC BY 4.0 and the source code under MIT.

Conformance and Usage

Before working with the standard, familiarise yourself with the terminology used throughout and how to approach completing a document.

Terminology

The key words “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “MAY”, and “OPTIONAL” in this standard are to be interpreted as described in RFC 2119 and RFC 8174. These RFCs define the meaning of requirement keywords used in technical standards.

Keyword	Meaning
SHALL	An absolute requirement of the standard
SHALL NOT	An absolute prohibition
SHOULD	Recommended; there may be valid reasons to deviate, but the architect must understand the implications
SHOULD NOT	Not recommended; there may be valid reasons to include but the implications must be understood
MAY / **OPTIONAL**	Truly optional; include at the author’s discretion

These keywords map to the standard’s documentation depths as follows:

Documentation Depth	Keyword	Governance Gate
[Minimum]	SHALL	Development / Test review
[Recommended]	SHOULD	Production approval
[Comprehensive]	MAY	Enterprise review

Acronyms used throughout the standard

ADS-specific acronyms:

Acronym	Expansion	Where it appears
ADS	Architecture Description Standard	This standard itself
SAD	Solution Architecture Document	The document you produce

Acronym	Expansion	Where it appears
HLD	High Level Design	The architectural-views and quality-attributes content within a SAD
ADR	Architecture Decision Record	Section 3.6.2 — captures the context, options, and consequences of a significant design decision
RAID	Risks, Assumptions, Issues, Dependencies	Section 6 — the four governance registers tracked across the SAD's life
ARB	Architecture Review Board	An organisation's governance forum that reviews and approves architectures

Common industry acronyms used in examples, templates, and prompts (defined here so new readers can follow without leaving the page):

Acronym	Expansion	Context
IaaS	Infrastructure as a Service	Cloud service model — provider supplies virtualised compute, storage, and networking; consumer manages everything above (OS, runtime, app)
PaaS	Platform as a Service	Cloud service model — provider supplies the runtime/platform; consumer manages only the application and data
SaaS	Software as a Service	Cloud service model — provider supplies the application; consumer manages only their data and configuration
RBAC	Role-Based Access Control	Authorisation model — permissions are granted to roles, and users are assigned to roles
GDPR	General Data Protection Regulation	EU/UK regulation governing personal data; relevant to Section 3.4 Data View and Section 2.3 Compliance

Getting Started

Use the workflow below to complete a Solution Architecture Document. Each step builds on the previous one.

Step 1 — Choose your depth. Pick the documentation depth for your project: - **[Minimum]** — Early-stage designs, proofs of concept, dev/test reviews - **[Recommended]** — Production-

bound designs requiring governance approval - **[Comprehensive]** — Critical, regulated, or enterprise-scale systems

Step 2 — Set the context (Sections 0–2). Start with document control, the executive summary, and stakeholders. This frames everything that follows.

Step 3 — Describe the architecture (Section 3). Work through each architectural view. A common starting order is the Logical View (what the solution does), then Physical View (where it runs), then the remaining views.

Step 4 — Assess quality (Section 4). Evaluate the design against each quality attribute. Document any tradeoffs between them.

Step 5 — Plan the lifecycle (Section 5). Document how the solution is built, deployed, operated, and eventually retired.

Step 6 — Capture decisions and risks (Section 6). Document constraints, assumptions, risks, dependencies, and issues. Record key architecture decisions.

Step 7 — Complete appendices and submit (Section 7). Add glossary, references, and approval sign-off. Complete the compliance scoring table and submit for governance review.

What to include and what to skip

- **Omit** sections that require a *higher* documentation depth than your target. For example, if your target is Recommended, omit sections marked Comprehensive. Do not include higher-depth sections with empty or "N/A" content.
- **Omit** sections at or below your target depth that are genuinely not relevant to the solution (e.g., IoT Devices for a cloud-only API).
- **Only include** sections that add value to the reader.

Document Lifecycle

A Solution Architecture Document produced to this standard is a **living artefact**:

- It SHALL describe the **current state** of the solution architecture
- The SAD SHALL be updated when architecturally significant changes are made
- It MAY be updated by multiple teams or change activities, either sequentially or in parallel
- It SHALL NOT be owned by a single project team; ownership rests with the accountable architect for the solution
- Changes to the design SHOULD be approved through the organisation's architecture governance process

Organisation Customisation

This standard is designed to be adopted without modification. However, organisations MAY extend it in the following ways:

1. **Organisation Profile** — Map generic sections to specific internal tools, standards, and governance processes using the `organisationProfile` field in the JSON Schema (a machine-readable definition of the document structure)
2. **Custom Sections** — Add organisation-specific sections using the `customSections` extension point (a way to add organisation-specific sections without modifying the core standard)

3. **Standards Traceability** — Reference internal design principles, patterns, and standards within the compliance traceability section (Section 6.8)
4. **Governance Gate Mapping** — Map the three documentation depths to your organisation's architecture governance stages (e.g., ARB, design authority)

Organisations SHALL NOT renumber core sections of the standard. Sections that are not applicable to the solution SHOULD be omitted entirely to keep documents focused and readable.

Output Formats

The standard's JSON Schema enables generation and validation in multiple formats:

Format	Use Case
Web	Interactive documentation site (this site)
Markdown	Inclusion in git repositories alongside code
YAML	Human-readable structured data, ideal for version control and configuration
JSON	Machine-readable for tooling integration and validation
Word / DOCX	Formal governance submissions and offline review (generated from Markdown)

See Templates for downloadable blank templates in each format.

Architecture Compliance Scoring

Each major section of a Solution Architecture Document MAY be assessed using a **0–5 compliance score**. This provides governance boards with a quick, quantitative view of architecture maturity and completeness — both for individual SADs and across a portfolio.

Scoring Scale

Score	Level	Description
0	Not Addressed	No evidence or content provided for this area
1	Acknowledged	The concern is recognised but no design or evidence exists
2	Partial	Some requirements addressed; significant gaps remain
3	Mostly Addressed	Most requirements met; minor gaps or risks requiring mitigation
4	Fully Addressed	All requirements met with supporting evidence
5	Exemplary	Best-practice implementation; could serve as a reference example

Where Scoring Applies

Scoring is applied at the **section level**, not at individual field level. The following sections are scored:

Section	Scoring Area	What 4/5 Looks Like
1. Executive Summary	Completeness & clarity	Clear business context, strategic alignment demonstrated, scope well-defined
3.1 Logical View	Design quality	Components well-decomposed, patterns justified, vendor lock-in assessed
3.2 Integration & Data Flow	Integration maturity	All interfaces documented, protocols specified, authentication defined
3.3 Physical View	Infrastructure rigour	Deployment architecture complete, environments specified, connectivity documented
3.4 Data View	Data governance	Data stores classified, retention defined, sovereignty addressed, encryption specified
3.5 Security View	Security posture	Threat model complete, all controls documented, monitoring in place
3.6 Scenarios	Design validation	Key use cases documented, ADRs capture significant decisions with rationale
4.1 Operational Excellence	Operational readiness	Centralised logging, monitoring, alerting, and runbooks all in place
4.2 Reliability	Resilience maturity	DR strategy defined, RTO/RPO met, backup tested, fault tolerance designed
4.3 Performance	Performance assurance	Targets defined, growth projected, testing approach documented
4.4 Cost Optimisation	Cost awareness	Cost analysis performed, monitoring enabled, right-sizing evidenced
4.5 Sustainability	Environmental consideration	Carbon-aware hosting, auto-shutdown, right-sizing
5. Lifecycle	Operational maturity	CI/CD documented, migration planned, skills assessed, exit plan in place
6. Decision Making	Governance rigour	RAID log complete, ADRs captured, compliance traceability demonstrated

Compliance Summary Table

Each SAD SHOULD include a compliance summary in the appendices (Section 7) or document control (Section 0):

Section	Score (0–5)	Assessor	Date	Notes
1. Executive Summary				
3.1 Logical View				
3.2 Integration & Data Flow				
3.3 Physical View				
3.4 Data View				
3.5 Security View				
3.6 Scenarios				
4.1 Operational Excellence				

Section	Score (0–5)	Assessor	Date	Notes
4.2 Reliability				
4.3 Performance				
4.4 Cost Optimisation				
4.5 Sustainability				
5. Lifecycle				
6. Decision Making				
Overall				

Guidance

- The **overall score** is typically the lowest individual section score (weakest-link principle), not an average. A solution with a 5 in performance but a 1 in security is not a "3" — it has a critical security gap.
- Scoring is **collaborative**, not adversarial. The solution architect and reviewing authority should score jointly (e.g., ARB or design authority).
- Scores below **3** on any section SHOULD trigger a remediation plan with clear ownership and timelines.
- Organisations MAY define minimum acceptable scores per section for different governance gates (e.g., "all sections must score ≥ 3 for production approval").

Normative References

See the full normative references table on the Framework Alignment page.

Design Principles

Atomic Field Design

ADS is designed with **atomic fields** — structured, discrete values rather than free-text wherever possible. This is a core principle of the standard.

Why Atomic Fields?

Benefit	Description
Reduced ambiguity	Authors select from defined options rather than inventing prose
Machine-processable	Tooling can aggregate, compare, validate, and report across SADs
Faster to complete	Checkboxes and selections are quicker than blank text fields
Comparable	Two SADs can be structurally compared and diffed
Translatable	Enum values can be mapped to any language without ambiguity
Governance-ready	The architecture governance process can verify completeness programmatically

How It Works Across Formats

The same atomic field renders differently depending on the output format, but carries identical meaning:

Schema Type	JSON / YAML	Markdown / Word	Confluence / Wiki
`boolean`	`true` / `false`	Checkbox: ☐ / ☑	Checkbox macro
`enum` (single)	`"active-active"`	Radio button or bold selection	Status macro or dropdown
`enum` (array)	`["aws", "azure"]`	Multiple checkboxes: ☑ AWS ☑ Azure	Multi-select macro
`yesNoNa`	`"yes"` / `"no"` / `"not-applicable"`	☑ Yes ☐ No ☐ N/A	Status macro
`riskLevel`	`"high"`	**HIGH** (with colour)	Status lozenge
`integer` / `number`	`4096`	Numeric field	Numeric field
`string` (constrained)	`"PT4H"` (ISO 8601)	“4 hours”	Duration field

Schema Type	JSON / YAML	Markdown / Word	Confluence / Wiki
`string` (free-text)	Prose	Prose	Prose

Atomic Field Categories in ADS

The schema uses the following atomic patterns:

Pattern	Example Fields	Values
Yes / No / N/A	`internetFacing`, `thirdPartyHosted`, `backupEnabled`	`yes`, `no`, `not-applicable`
Risk Level	`residualRisk`, `confidentiality`, `integrity`	`critical`, `high`, `medium`, `low`, `negligible`
Classification	`classification`	`public`, `internal`, `restricted`, `highly-restricted`
Status	`document status`, `ADR status`, `risk status`	Defined per context
Technology Enum	`protocol`, `authenticationMethod`, `secretStore`	Defined per field
Sizing Category	`dataSizeCategory`, `clusterSize`	Banded ranges
Duration	`retentionPeriod`, `rtoTarget`	ISO 8601 or banded categories
Criticality	`businessCriticality`	`tier-1-critical` through `tier-5-minimal`
ID Pattern	`UC-01`, `ADR-001`, `R-001`	Regex-constrained strings

Multi-Format Compatibility

ADS is designed to work equally well as:

- **A human-authored document** (Word, Confluence, wiki) — where atomic fields become checkboxes, dropdown menus, and radio buttons
- **A machine-readable file** (JSON, YAML, Markdown) — where atomic fields are enums, booleans, and typed values
- **A web form** — where atomic fields map directly to form controls

The JSON Schema is the single source of truth. All output formats are derived from it, ensuring consistency regardless of how the SAD is authored or consumed.

Accessibility & Cognitive Load

ADS is designed to be usable by all architects, including those who are neurodivergent. The following principles reduce cognitive load and improve readability:

Structural Consistency

Every section in the standard follows the same pattern:

1. **Purpose** — a short statement explaining what this section covers and why
2. **Tables / Fields** — structured, atomic fields to complete
3. **Guidance** — contextual help in a clearly marked box

This predictable structure means you always know what to expect when you open a new section.

Reducing Cognitive Load

Principle	How ADS Implements It
Atomic fields over blank pages	Checkboxes, enums, and structured tables replace open-ended text fields. You select rather than compose.
Progressive disclosure	Three documentation depths (Minimum → Recommended → Comprehensive) let you start simple and add detail only when needed.
One idea per row	Tables present one field per row rather than combining multiple concepts.
Consistent terminology	The glossary defines all acronyms. Terms are used consistently throughout.
Visual markers	Colour-coded depth indicators (**[Minimum]** **[Recommended]** **[Comprehensive]**) let you quickly see what's required.
Clear scoring	The 0–5 compliance scale provides unambiguous, numeric feedback rather than subjective prose.

Acronyms & Terminology

All acronyms used in ADS are defined in the glossary. When completing a SAD:

- Define all acronyms on first use within the document
- Include a glossary (Section 7.1) in every SAD
- Avoid jargon where a plain-language alternative exists

Readability Guidelines for SAD Authors

When writing free-text content in a SAD:

- Use short sentences. One idea per sentence.
- Use bullet points rather than long paragraphs.
- Use active voice ("The service sends data to…") not passive ("Data is sent by the service to…").
- Use diagrams to complement text, not replace it.
- Ensure diagrams have alt-text descriptions for screen readers.

Sustainability as a First-Class Concern

ADS treats sustainability as a quality attribute on equal footing with reliability, performance, and cost — not as an afterthought or a one-line tick-box.

Principle	How ADS Implements It
A dedicated quality attribute	Section 4.5 covers hosting efficiency, carbon measurement, network efficiency, code efficiency, and data efficiency, with prompts and a 0–5 scoring scale.
Embedded in the views, not bolted on	The Physical View (3.3), Data View (3.4), and Logical View (3.1) each have explicit *Sustainability Considerations* prompts at the point where the carbon-impacting decisions are actually made.
Operational, not aspirational	The Lifecycle Management section (5.5) requires documenting how non-prod auto-shutdown, right-sizing reviews, and orphan-resource reclamation are operationalised — not just intended.
Aligned to recognised frameworks	Cross-references to AWS Well-Architected Sustainability and the Green Software Foundation's Software Carbon Intensity (SCI) specification.
Trade-offs documented, not buried	Where sustainability conflicts with another quality attribute (e.g., always-on DR vs cold standby), the trade-off is captured in the Quality Attribute Trade-offs section of the Section 4 overview rather than left implicit.

The intent is that a SAD reviewer who reads only the views and the lifecycle section, never opening Section 4.5, still encounters the carbon-impacting decisions and can challenge them.

Extensibility Without Ambiguity

When organisations add custom sections via the `customSections` extension point, they SHOULD follow the same atomic field principles:

1. Use `boolean` for yes/no questions
2. Define `enum` values for any field with a finite set of valid answers
3. Use typed fields (`integer`, `number`, `date`) for measurable values
4. Reserve free-text `string` fields for genuinely open-ended content
5. Add `description` to every field to clarify its purpose

Framework Alignment

Alignment Model

ADS maps to multiple established frameworks. This page documents both the alignment (what ADS shares with each framework) and the differences (what ADS adds or does differently).

Framework Badges

Each section page in the standard shows badges indicating which frameworks influenced that section:

Badge	Meaning
`ISO 42010`	Aligns with the ISO/IEC/IEEE 42010 conceptual model
`4+1 View`	Maps to a view in Kruchten's 4+1 Architectural View Model
`TOGAF`	Aligns with a TOGAF architecture domain
`AWS WAF`	Derived from the AWS Well-Architected Framework
`Azure WAF`	Derived from the Azure Well-Architected Framework

The badges are informational only — they show provenance, not requirements. You do not need to comply with these external frameworks to conform to ADS.

ISO/IEC/IEEE 42010

What ADS Shares with ISO 42010

ADS follows the ISO 42010 conceptual model for architecture descriptions:

ISO 42010 Concept	ADS Mapping
System of Interest	The solution being documented
Stakeholders	Stakeholders & Concerns
Concerns	Concerns Matrix
Architecture Viewpoint	Defined viewpoints
Architecture View	The content within each view
Model Kind	Diagrams, tables, and descriptions within views
Correspondences	Cross-references between views and quality attribute refs

ISO 42010 Concept	ADS Mapping
Architecture Description	The complete Solution Architecture Document

Where ADS Goes Beyond ISO 42010

ISO 42010 is a meta-standard — it defines *concepts* but deliberately leaves the concrete structure to implementers. ADS provides that concrete structure:

Aspect	ISO 42010	ADS
Prescribes sections?	No — defines concepts only	Yes — exact sections, tables, and fields
Quality assessment?	No	Yes — quality attributes from cloud WAFs
Documentation depths?	No	Yes — Minimum, Recommended, Comprehensive with RFC 2119
Security viewpoint?	Not prescribed	Yes — dedicated Security View
Lifecycle coverage?	Focuses on the AD itself	Yes — CI/CD, operations, migration, exit planning
Risk and governance?	No	Yes — constraints, risks, assumptions, ADRs, compliance
Templates?	No	Yes — Markdown, YAML, JSON
Machine-readable schema?	No	Yes — JSON Schema with validation

Kruchten 4+1 View Model

What ADS Shares with 4+1

ADS uses the 4+1 model as the foundation for its architectural views:

4+1 Original View	ADS View	Key Content
Logical View	Logical View	Component architecture, service mapping
Process View	Integration & Data Flow View *(adapted)*	Data flows, integrations, interfaces
Development View	Lifecycle Management	CI/CD, SDLC, tooling

4+1 Original View	ADS View	Key Content
Physical View	Physical View	Deployment, infrastructure, networking
+1 Scenarios	Scenarios	Use cases, ADRs

Where ADS Extends 4+1

Aspect	4+1	ADS
Data architecture?	Cross-cutting across views	Dedicated Data View
Security architecture?	Cross-cutting across views	Dedicated Security View
Process View scope	Concurrency and threading	Adapted to integration and data flows
Quality assessment?	No	Yes — cross-cutting quality attributes
Documentation depths?	No	Yes — three tiers

Cloud Well-Architected Frameworks (AWS, Azure, GCP, Oracle, IBM)

What ADS Shares with Cloud WAFs

ADS derives its quality attributes from the cloud Well-Architected Frameworks:

Note: The cloud providers use American English spelling (e.g., "Optimization"). ADS uses British English (e.g., "Optimisation"). The table below preserves each provider's original spelling.

Quality Attribute	AWS	Azure	GCP	Oracle	IBM
Operational Excellence	Operational Excel-lence	Operational Excellence	Operational Excel-lence	Operational Efficiency	Operational Excel-lence
Security	Security *(ADS maps this to a dedic-ated Security View)*	Security *(ADS maps this to a dedicated Security View)*	Security, Privacy & Com-pliance *(ADS maps this to a dedic-ated Security View)*	Security & Compliance *(ADS maps this to a dedicated Security View)*	Security *(ADS maps this to a dedic-ated Security View)*
Reliability	Reliability	Reliability	Reliability	Reliability & Resilience	Reliability

Quality Attribute	AWS	Azure	GCP	Oracle	IBM
Performance	Performance Efficiency	Performance Efficiency	Performance Optimization	*(combined with Cost)*	Performance
Cost	Cost Optimization	Cost Optimization	Cost Optimization	Performance & Cost Optimization	Cost Optimization
Sustainability	Sustainability	*(not separate)*	Sustainability	*(not separate)*	*(not separate)*
(provider-specific)	—	—	—	Distributed Cloud	Hybrid & Portable

Where ADS Differs from Cloud WAFs

Aspect	Cloud WAFs	ADS
Focus	Quality assessment only	Full SAD structure (views + quality + lifecycle + governance)
Architectural views?	No	Yes — six views
Security	A quality pillar	Elevated to a dedicated Architectural View
Document structure?	No — assessment questions only	Yes — prescriptive sections and tables
Templates?	No	Yes — Markdown, YAML, JSON
Provider-neutral?	Each is provider-specific	Yes — maps across all providers

TOGAF

What ADS Shares with TOGAF

ADS maps its sections to TOGAF's four architecture domains:

TOGAF Domain	ADS Section
Business Architecture	Executive Summary, Stakeholders
Data Architecture	Data View
Application Architecture	Logical View, Integration & Data Flow View
Technology Architecture	Physical View

Where ADS Differs from TOGAF

Aspect	TOGAF	ADS
Scope	Full enterprise architecture framework (ADM lifecycle, capability planning, governance)	SAD template only — focused on documenting a single solution
Prescribes SAD sections?	No — TOGAF describes what an architecture should *address* but not the document structure	Yes — exact sections, tables, and fields
Quality attributes?	No	Yes — cross-cutting quality attributes from cloud WAFs
Security view?	Security is a cross-cutting concern	Yes — dedicated Security View
Templates?	No (paid ArchiMate tooling available separately)	Yes — free Markdown, YAML, JSON templates
Compliance scoring?	No	Yes — 0–5 scale per section
Cost	Paid (TOGAF certification and library access)	Free and open-source (CC BY 4.0)

arc42

What ADS Shares with arc42

Both ADS and arc42 aim to provide a practical template for documenting software/solution architecture. They share: - A view-based approach to describing architecture - Explicit sections for quality requirements - A glossary and cross-references section

Where ADS Differs from arc42

Aspect	arc42	ADS
Security view?	No — security is in "cross-cutting concepts"	Yes — dedicated Security View
Data view?	No	Yes — dedicated Data View
Quality attributes?	Quality scenarios (unstructured)	Structured quality attributes aligned to cloud WAFs
Documentation depths?	No	Yes — three tiers with RFC 2119
Machine-readable schema?	No — prose only	Yes — JSON Schema with validation
Compliance scoring?	No	Yes — 0–5 scale per section

Aspect	arc42	ADS
Governance section?	Partial (risks and technical debt)	Full — constraints, assumptions, risks, dependencies, issues, ADRs, compliance traceability
Lifecycle management?	Partial	Full — CI/CD, migration, resourcing, decommissioning, exit planning

Section-to-Framework Traceability Matrix

ADS Section	ISO 42010	4+1	Cloud WAFs	TOGAF
Document Control	AD metadata	—	—	—
Executive Summary	—	—	—	Business
Stakeholders	Stakeholders, Concerns	—	—	Business
Logical View	Viewpoint, View	Logical	—	Application
Integration & Data Flow	Viewpoint, View	Process *(adapted)*	—	Application
Physical View	Viewpoint, View	Physical *(extended)*	—	Technology
Data View	Viewpoint, View	—	—	Data
Security View	Viewpoint, View	*(extended)*	Security	—
Scenarios	Correspondences	+1 Scenarios	—	—
Operational Excellence	—	—	Ops Excellence	—
Reliability & Resilience	—	—	Reliability	—
Performance Efficiency	—	—	Performance	—
Cost Optimisation	—	—	Cost Optimization	—
Sustainability	—	—	Sustainability *(AWS, GCP)*	—
Lifecycle Management	—	Development	Ops Excellence	—
Decision Making & Governance	Correspondences	—	—	—

0. Document Control

[Minimum] `ISO 42010`

Purpose

This section captures the administrative metadata for the Solution Architecture Document, ensuring it is versioned, traceable, and governed appropriately.

0.1 Document Metadata

[Minimum]

Field	Description
Document Title	The formal title of this Solution Architecture Document
Application / Solution Name	The name of the system being documented
Application ID	Unique identifier (e.g., from a CMDB or portfolio tool)
Author(s)	Primary author(s) of the document
Owner	The accountable architect for this document
Version	Current document version (use semantic versioning)
Status	Draft / In Review / Approved / Superseded
Created Date	Date the document was first created
Last Updated	Date of the most recent update
Classification	Document security classification

0.2 Change History

[Minimum]

Track all significant changes to the document:

Version	Date	Author / Editor	Description of Change
0.1	*[date]*	*[name]*	Initial draft
...	...	...	...

Guidance

- Use semantic versioning: `MAJOR.MINOR` (e.g., 1.0 for first approved version, 1.1 for minor updates, 2.0 for significant redesign)
- Record who made each change and why

- This is a living document - it should be updated whenever architecturally significant changes are made to the solution

0.3 Contributors & Approvals

[Recommended]

Tracking contributors and their roles ensures accountability and provides an audit trail for governance.

Name	Role	Contribution Type
[name]	*[role]*	Author / Reviewer / Approver

0.4 Document Purpose & Scope

[Minimum]

Describe the purpose of this specific Solution Architecture Document:

- State which solution or application this document describes.
- Define the scope boundary.
- What is explicitly out of scope?
- What other documents does it relate to (e.g., detailed designs, operational runbooks)?

Guidance

The Solution Architecture Document serves as: - A **reference document** describing the current-state solution architecture - **Evidence** that the solution is well-architected - **Input** to architecture governance and review processes - A **living artefact** that must be updated when architecturally significant changes are made

It should contain design details that are relatively **static**. Dynamic or frequently changing information belongs in operational documentation.

1. Executive Summary

[Minimum] `TOGAF`

Purpose

The Executive Summary provides a concise overview of the solution for all stakeholders, including non-technical decision-makers. It establishes the business context and frames the architectural decisions that follow.

1.1 Solution Overview

[Minimum]

Provide a brief (1-2 paragraph) summary of the solution:

- Describe what the solution does.
- Describe the business problem it solves.
- Describe the high-level approach.

[...]

1.2 Business Context & Drivers

[Minimum]

Describe the business drivers that led to this solution:

Driver	Description	Priority
[business driver]	*[description]*	High / Medium / Low

Consider: - Business requirements and objectives - Regulatory or compliance drivers - Technology modernisation goals - Cost reduction or efficiency targets - Market or competitive pressures

1.3 Strategic Alignment

[Recommended]

Demonstrate how this solution aligns with organisational strategy and avoids duplication of existing capabilities.

Organisational Strategy Alignment

Question	Response
Which organisational strategy or initiative does this solution support?	*[e.g., digital transformation programme, cloud-first strategy]*
Has this solution been reviewed against the organisation's capability model?	Yes / No / N/A
Does this solution duplicate any existing capability?	Yes / No — *[if yes, provide justification]*

Reuse of Shared Services & Platforms

Document which existing organisational platforms, shared services, or common components are being reused — and which were considered but not selected (with justification).

Capability	Shared Service / Platform	Reused?	Justification (if not reused)
Identity & Access	*[e.g., Entra ID, Okta]*	Yes / No	*[...]*
Messaging / Notifications	*[e.g., GOV.UK Notify, SES, SendGrid]*	Yes / No	*[...]*
API Management	*[e.g., Apigee, AWS API Gateway, Kong]*	Yes / No	*[...]*
Monitoring & Logging	*[e.g., Splunk, Datadog, Grafana]*	Yes / No	*[...]*
Data & Analytics	*[e.g., Snowflake, Databricks]*	Yes / No	*[...]*
CI/CD	*[e.g., GitHub Actions, Jenkins, Azure DevOps]*	Yes / No	*[...]*
Other	*[...]*	Yes / No	*[...]*

Guidance

Most organisations maintain a catalogue of approved platforms and shared services. Architects should demonstrate they have assessed these before proposing new technology. This reduces cost, complexity, and operational overhead. If a shared service is not reused, provide a clear justification. Common reasons include: - Capability gap — the shared service does not meet requirements - Performance — the shared service cannot meet performance targets - Contractual — licensing or vendor constraints prevent use

1.4 Scope

[Minimum]

In Scope

Define what this SAD covers:

- Which applications, services, or components?

- Which environments (production, DR, test)?
- Which user groups or business functions?

Out of Scope

Explicitly state what is excluded. Examples may include:

- Related systems documented elsewhere
- Future phases not covered in this design
- Detailed designs (covered in separate documents)
- Infrastructure managed by other teams
- Third-party systems outside the solution boundary

1.5 Current State / As-Is Architecture

[Recommended]

Understanding the current state helps stakeholders assess the scale of change and identify transition risks.

Where the solution replaces, modifies, or integrates with existing systems, describe the current-state architecture that is relevant to the change:

- What existing systems or components are affected?
- What are the key limitations or pain points of the current state?
- What is being retained, replaced, or modified?

[Insert current-state architecture diagram if applicable]

Guidance

This section is particularly valuable for migration and modernisation projects. For greenfield solutions with no existing landscape, this section SHOULD be omitted. Keep the description focused on what is directly relevant to the transition — do not reproduce extensive legacy documentation that exists elsewhere.

1.6 Key Decisions & Constraints

[Minimum]

Summarise the most significant architectural decisions and any constraints that shaped the design:

Decision / Constraint	Rationale	Impact
[decision]	*[why]*	*[effect on design]*

Guidance

Constraints may include: - **Technical** - Existing technology stack, platform mandates - **Organisational** - Team capabilities, vendor relationships - **Financial** - Budget limits, cost targets - **Regulatory** - Compliance requirements, data residency - **Time** - Delivery deadlines, phased rollouts

1.7 Project Details

[Recommended]

Linking the architecture to a specific project or change activity provides traceability for governance.

If this SAD is produced as part of a specific project or change activity:

Field	Value
Project Name	*[...]*
Project Code / ID	*[...]*
Project Manager	*[...]*
Estimated Solution Cost (Capex)	*[...]*
Estimated Solution Cost (Opex)	*[...]*
Target Go-Live Date	*[...]*

1.8 Business Criticality

[Recommended]

Business criticality drives the level of resilience, recovery, and governance rigour required throughout the design.

Classify the business criticality of the solution. The chosen tier suggests a recommended documentation depth:

Tier	Description	Recommended Depth
Tier 1: Critical	Service failure causes immediate, severe business impact	**[Comprehensive]**
Tier 2: High Impact	Service failure significantly impacts business operations	**[Comprehensive]**
Tier 3: Medium Impact	Service failure impacts operations but workarounds exist	**[Recommended]**
Tier 4: Low Impact	Service failure has limited business impact	**[Recommended]**
Tier 5: Minimal	Service failure has negligible impact	**[Minimum]**

Selected criticality: *[...]*

This classification drives requirements for resilience, recovery time objectives, and governance rigour throughout the design. See the Depth Cheat Sheet for what to fill in at each level.

Scoring Guidance

Score	What This Looks Like
1	Business context mentioned but vague; scope undefined

Score	What This Looks Like
3	Clear business drivers, scope defined, strategic alignment documented
5	All of the above plus current-state architecture documented, reuse assessment complete, business criticality justified with evidence

2. Stakeholders & Concerns

[Recommended] `ISO 42010` `TOGAF`

Purpose

ISO/IEC/IEEE 42010 requires that an architecture description identify stakeholders and their concerns. This section ensures the SAD addresses the needs of all relevant parties and documents the compliance and regulatory context.

2.1 Stakeholder Register

[Minimum]

Identify all stakeholders with an interest in the solution's architecture:

Stakeholder	Role / Group	Key Concerns	Relevant Views
Business Owner	Business	Business value, cost, timeline	Executive Summary, Cost
Solution Architect	Architecture	Design integrity, standards compliance	All views
Security Architect	Security	Threat model, access control, data protection	Security View
Infrastructure Engineer	Operations	Deployment, scaling, networking	Physical View
Data Architect	Data Management	Data storage, classification, privacy	Data View
Development Lead	Development	Component design, integration patterns	Logical View, Integration & Data Flow View
Operations / SRE	Operations	Observability, incident response, reliability	Quality Attributes
Compliance Officer	Compliance	Regulatory adherence, audit evidence	Governance
[additional stakeholders]	*[role]*	*[concerns]*	*[views]*

Guidance

Consider stakeholders from: - **Business** - sponsors, product owners, end users - **Technology** - architects, engineers, developers, DBAs - **Operations** - SRE, support teams, NOC - **Security & Compliance** - CISO office, risk, audit - **External** - vendors, customers, regulators, partners

2.2 Concerns Matrix

[Recommended]

Map stakeholder concerns to the views and sections that address them:

Concern	Stakeholder(s)	Addressed In
Solution meets business requirements	Business Owner	1. Executive Summary, 3.6 Scenarios
Solution is secure and compliant	Security Architect, Compliance	3.5 Security View, 6. Governance
Solution is reliable and recoverable	Operations, Business	4.2 Reliability & Resilience
Solution is cost-effective	Business Owner, Finance	4.4 Cost Optimisation
Solution can be operated and monitored	Operations / SRE	4.1 Operational Excellence
Data is properly managed and protected	Data Architect, Compliance	3.4 Data View
Solution can scale to meet demand	Infrastructure Engineer	4.2 Reliability, 3.3 Physical View
[additional concerns]	*[stakeholders]*	*[sections]*

Guidance

The concerns matrix ensures every stakeholder's key concerns are traceable to specific sections of the SAD. This helps reviewers verify that the architecture addresses all stakeholder needs and helps authors understand which sections matter most to which audience.

2.3 Compliance & Regulatory Context

[Recommended]

Document the regulatory and compliance landscape that applies to this solution:

Regulatory Requirements

Regulation / Standard	Applicability	Impact on Design
[e.g., UK GDPR, PCI-DSS, US SOX, UK FCA]	*[how it applies]*	*[design implications]*

Regulated Activities

Does the solution support any regulated activities?

☐ Yes - *[describe which regulated activities and entities]*
☐ No

Compliance Standards

List any internal or external standards that the design must conform to:

Standard	Version	Applicability
[e.g., internal security standard]	*[version]*	*[which sections]*

Guidance

Identifying the compliance landscape early shapes the entire design. Common regulations to consider: - **Data protection** — UK GDPR, EU GDPR, UK Data Protection Act, US CCPA - **Financial services** — PCI-DSS, US SOX, UK FCA rules, EU PSD2 - **Healthcare** — NHS DSPT (UK), US HIPAA, HL7/FHIR standards - **Security** — ISO 27001, NIST CSF (US), Cyber Essentials (UK), SOC 2 - **Internal** — organisational security policies, cloud platform standards, data classification policies

3. Architectural Views

`ISO 42010 4+1 Views`

Purpose

This page introduces the architectural views and explains how they relate to each other. Each view is detailed in the sub-pages that follow.

Now that the stakeholders and their concerns are identified (Section 2), the architectural views describe the solution that addresses those concerns. Each view speaks to different stakeholder groups.

View Model

The ADS organises the solution architecture into **six complementary views**, each addressing a distinct set of stakeholder concerns. This approach is aligned with Kruchten's 4+1 Architectural View Model, extended with dedicated Data and Security views.

No single view provides a complete picture of the architecture. Together, the views provide a holistic description that can be understood by all stakeholders.

The Views

#	View	Based On	Primary Stakeholders	Describes
3.1	**Logical View**	4+1 Logical	Architects, Developers	Application components, services, patterns
3.2	**Integration & Data Flow View**	4+1 Process (adapted)	Integrators, Architects	Data flows, integrations, interfaces
3.3	**Physical View**	4+1 Physical	Infrastructure, DevOps, Cloud	All deployment infrastructure — physical, virtual, cloud, and serverless
3.4	**Data View**	RM-ODP Information	Data Architects, DBA, Compliance	Data stores, classification, privacy, lifecycle

#	View	Based On	Primary Stakeholders	Describes
3.5	**Security View**	Extended	Security, CISO, Compliance	IAM, encryption, monitoring, threat model
3.6	**Scenarios**	4+1 +1	All Stakeholders	Use cases, architecture decision records

Correspondences Between Views

Views are not independent - they share correspondences (relationships between elements across views). Key correspondences include:

- **Logical components** (3.1) are **deployed onto** physical infrastructure (3.3)
- **Data stores** (3.4) are **hosted within** the physical environment (3.3)
- **Integration flows** (3.2) traverse **network paths** defined in the physical view (3.3)
- **Security controls** (3.5) are **applied to** components, data, and infrastructure across all views
- **Scenarios** (3.6) **validate** the design across all views

When documenting a view, cross-reference related elements in other views to maintain traceability.

Diagram Conventions

Each view should include at least one diagram. Follow these conventions:

- Use a consistent notation (e.g., UML, ArchiMate, C4 Model, or simple block diagrams)
- Label all components, connections, and boundaries clearly
- Show trust boundaries in security-relevant diagrams
- Include a legend if notation is not self-evident
- Ensure diagrams are at a resolution that allows text to be read

3.1 Logical View

[Minimum] `4+1 Logical TOGAF Application`

Purpose

The Logical View describes the solution's functional structure - the application components, how they relate to each other, and how they map to business services and capabilities. It addresses the concerns of solution architects and developers.

3.1.1 Application Architecture Diagram

[Minimum]

Provide a block diagram showing the software components used in the solution (applications, middleware, databases, APIs, etc.), how they interact, and how they integrate with external platforms.

[Insert application architecture diagram]

Guidance

The diagram should show: - All major application components and their roles - Interactions and data flows between components - Integration points with external systems - Clear boundaries between the solution and its environment - Consider using the C4 Model (a hierarchical approach to diagramming: Context, Container, Component, Code) for progressive detail

3.1.2 Component Decomposition

[Recommended]

Describe each major component of the solution:

Component	Type	Description	Technology	Owner
[name]	Application / Service / API / Database / Queue / etc.	*[what it does]*	*[technology stack]*	*[team]*

3.1.3 Service & Capability Mapping

[Recommended]

Mapping components to business capabilities helps identify duplication and ensures the solution aligns with the organisation's capability model.

Map the solution to business services and capabilities it supports:

Service ID	Service Name	Capability ID	Capability Name
[...]	*[...]*	*[...]*	*[...]*

3.1.4 Application Impact

[Recommended]

Tracking impacted applications ensures that downstream teams are informed and that integration testing covers all affected systems.

Identify other applications that are impacted by this solution:

Application Name	Application ID	Impact Type	Change Details	Comments
[name]	*[id]*	Use / Change / Create	*[details]*	*[notes]*

3.1.5 Key Design Patterns

[Comprehensive]

Document the architectural and design patterns applied in this solution:

Pattern	Where Applied	Rationale
[e.g., Microservices, Event-Driven, CQRS, Strangler Fig]	*[which components]*	*[why this pattern was chosen]*

3.1.6 Technology & Vendor Lock-in Assessment

[Recommended]

Assess the degree of vendor or technology lock-in for key components:

Component / Service	Vendor / Technology	Lock-in Level	Mitigation	Portability Notes
[component]	*[vendor/technology]*	None / Low / Moderate / High / Critical	*[how lock-in is mitigated]*	*[how portable is this component?]*

Guidance

Consider lock-in across multiple dimensions: - **Proprietary APIs** — Does the solution use vendor-specific APIs with no open equivalent? - **Data formats** — Can data be exported in standard, open formats? - **Contractual** — Are there minimum commitment periods or exit penalties? - **Operational** — Does the team have skills to operate on an alternative platform? - **Migration effort** — How much re-engineering would be needed to move to an alternative?

This assessment should inform the Exit Planning section (5.10) and the Cost Optimisation quality attribute (4.4).

3.1.7 Sustainability Considerations

[Comprehensive]

Component-level decisions influence the carbon profile of the running system. Capture the sustainability-relevant patterns; details and metrics belong in Section 4.5.

Question	Response
Where appropriate, has caching been used to avoid recomputation or repeated downstream calls?	Yes / No — *[which components, what's cached]*
Are batch processes consolidated rather than continuously polling?	Yes / No / Not applicable
Are async / event-driven patterns used to flatten peak load and right-size compute?	Yes / No — *[where]*
Have heavy framework choices been weighed against lighter alternatives where the workload allows?	Yes / No — *[discussion]*

Why this matters

The biggest sustainability gains in the Logical View are usually **avoiding work** rather than **doing work efficiently**: a cached response, a debounced poll, an event consumed once instead of polled fifty times. These are architectural patterns chosen here, not infrastructure tweaks made later.

Scoring Guidance

Score	What This Looks Like
1	High-level diagram exists but components not decomposed
3	Components documented with technology choices, design patterns justified
5	All of the above plus vendor lock-in assessed, component interactions fully specified, all statuses (new/existing/decommissioned) documented

Quality Attribute Cross-References: - **4.3 Performance Efficiency** - Component design affects performance characteristics - **4.2 Reliability** - Component decomposition influences fault isolation and resilience - **4.4 Cost Optimisation** - Vendor lock-in affects long-term cost flexibility - **4.5 Sustainability** - Component architecture affects resource utilisation

3.2 Integration & Data Flow View

[Minimum] `4+1 Process (adapted)`

Purpose

The Integration & Data Flow View describes how data moves through the solution, how components communicate, and how the solution integrates with internal and external systems. It addresses the concerns of integrators, operations teams, and architects.

3.2.1 Data Flow Diagrams

[Recommended]

Illustrate how data moves through the solution from input to output, showing transformations and processing stages.

[Insert data flow diagram(s)]

Guidance

Consider showing: - Data ingestion and sourcing - Processing and transformation steps - Data outputs and destinations - Asynchronous vs synchronous flows - Event-driven patterns and message queues

3.2.2 Internal Component Connectivity

[Minimum]

Document the connectivity between internal components of the solution:

Source Component	Destination Component	Protocol / Encryption	Authentication Method	Purpose
[component]	*[component]*	*[e.g., HTTPS/TLS 1.3]*	*[e.g., mTLS, API Key]*	*purpose*

Guidance

List all connections between the solution's internal components. For each connection, document: - Connection initiator (source) and receiver (destination) - Protocol and encryption standard used - Authentication and authorisation method - Whether the connection is synchronous or asynchronous

3.2.3 External Integration Architecture

[Minimum]

Document how the solution interfaces with external systems, third-party services, and other internal applications:

Source Application	Destination Application	Protocol / Encryption	Authentication	Security Proxy	Purpose
[app name / ID]	*[app name / ID]*	*[protocol]*	*[method]*	*[if applicable]*	*purpose*

End User Access

Document how end users access the solution:

User Type	Access Method	Authentication	Protocol
[e.g., Internal employees]	*[e.g., Web browser via SSO]*	*[e.g., SAML/OIDC]*	*[HTTPS]*
[e.g., External customers]	*[e.g., API]*	*[e.g., OAuth 2.0]*	*[HTTPS]*

3.2.4 API & Interface Contracts

[Comprehensive]

Document the key APIs and interfaces exposed or consumed by the solution:

API / Interface	Type	Direction	Format	Version	Documentation
[name]	REST / GraphQL / gRPC / SOAP / Event	Exposed / Consumed	JSON / XML / Protobuf	*[version]*	*[link]*

Scoring Guidance

Score	What This Looks Like
1	Data flow diagram exists but interfaces not detailed
3	All internal and external integrations documented with protocols, authentication, and direction
5	All of the above plus API contracts versioned, SLA/SLO defined per interface, user access patterns documented

Quality Attribute Cross-References: - **4.1 Operational Excellence** - Integration failures are a major source of incidents - **4.2 Reliability** - External dependencies affect reliability posture - **4.3 Performance** - Network latency and throughput in integration paths

3.3 Physical View

[Minimum] `4+1 Physical TOGAF Technology`

Purpose

The Physical View describes **all infrastructure** that hosts and supports the solution — whether physical hardware, virtual machines, containers, serverless functions, or cloud-managed services. It addresses the concerns of infrastructure engineers, DevOps teams, platform engineers, and cloud architects.

3.3.1 Deployment Architecture Diagram

[Minimum]

Provide a diagram showing the infrastructure that drives the solution.

[Insert deployment architecture diagram]

Guidance

Show: physical/virtual servers, containers, cloud services, storage, networks, firewalls, load balancers, Internet gateways, SaaS platforms, and any other infrastructure components. Include cloud regions, availability zones, and VPCs where applicable.

3.3.2 Hosting & Infrastructure

[Minimum]

Hosting Venues

[Minimum]

Attribute	Selection
Hosting Venue Type	Cloud / SaaS / On-Premises / Hybrid / Outsourced
Hosting Region(s)	*[e.g., UK, EU, US, Asia, Other]*
Service Model	IaaS / PaaS / SaaS / FaaS / Other
Cloud Provider	AWS / Azure / GCP / Other / N/A
Account / Subscription Type	*[organisation-specific account type]*

Compute

[Recommended]

Servers (Physical / Virtual)

Instance Name	Instance Type	vCPU	Memory (GB)	Storage	Quantity	OS
[name]	*[type/size]*	*[n]*	*[n]*	*[n TB]*	*[n]*	*[OS version]*

Containers

Attribute	Detail
Container Platform	EKS (AWS) / AKS (Azure) / GKE (GCP) / Docker / Kubernetes / Other
Base Image(s)	*[e.g., Alpine, Node, NGINX]*
Cluster Size	*[number of nodes]*

Serverless

Attribute	Detail
Serverless Services	*[e.g., AWS Lambda, Azure Functions]*
Function Details	*[description of serverless components]*

High Performance Computing

If the solution uses specialised compute:

Attribute	Detail
HPC Requirements	*[e.g., GPU, FPGA, specialised compute]*

Artificial Intelligence / Machine Learning

If the solution includes AI or machine learning components:

Attribute	Detail
AI/ML Components	*[training and inference infrastructure]*

Security Agents

[Recommended]

Document security software deployed on compute resources. This is captured in the Physical View because agents are infrastructure-level components, even though they serve a security function.

Document security agents deployed on compute resources:

☐ Anti-Malware
☐ Endpoint Detection and Response (EDR)
☐ Vulnerability Management
☐ Other: *[...]*

3.3.3 Network Topology & Connectivity

[Minimum]

Connectivity Summary

Question	Response
Is this an Internet-facing application?	Yes / No - *[details]*
Outbound Internet connectivity required?	Yes / No - *[details]*
Cloud-to-on-premises connectivity required?	Yes / No - *[details]*
Wireless networking required?	Yes / No - *[details]*
Third-party / co-location connectivity required?	Yes / No - *[details]*
Cloud network peering required?	Yes / No - *[details]*

User & Administrator Access

[Recommended]

Document how users and administrators connect to the solution, including access methods, protocols, and network connectivity.

Attribute	Selection
User access method	Web (HTTPS) / VDI / RDP / Citrix / Mobile App / API / Other
User locations	*[e.g., UK offices, Remote (VPN), Global, End-customers (Internet)]*
Administrator access method	VDI / RDP / SSH / HTTPS / Bastion Host / Other
VPN required	Yes / No
Direct Connect / ExpressRoute	Yes / No

Transport Protocols

[Recommended]

Documenting transport protocols helps security and network teams verify that all communication paths use appropriate encryption and authentication.

Protocol	Used?	Purpose
HTTPS (TLS 1.2+)	Yes / No	*[...]*
SFTP	Yes / No	*[...]*
ODBC / JDBC	Yes / No	*[...]*

Protocol	Used?	Purpose
TCP (other)	Yes / No	*[...]*
gRPC	Yes / No	*[...]*
WebSocket	Yes / No	*[...]*
Other	Yes / No	*[...]*

Network Bandwidth

[Recommended]

Bandwidth requirements inform infrastructure sizing and cost estimation. Underestimating can cause performance issues; overestimating wastes budget.

Metric	Value
Peak egress bandwidth to Internet	*[Mb/s]*
Peak ingress bandwidth from Internet	*[Mb/s]*
Peak bandwidth between on-prem and cloud	*[Mb/s]*
Traffic characteristics	*[constant / burst / periodic]*
QoS requirements	*[details]*
Network performance expectations	*[latency, jitter, etc.]*

Internet Perimeter Protection

[Recommended]

Control	Implemented	Detail
DDoS Protection	Yes / No	*[service used]*
Rate Limiting	Yes / No	*[details]*
Source IP Restrictions	Yes / No	*[IP allowlist, geo-blocking]*
Web Application Firewall (WAF)	Yes / No	*[product]*
Client Verification Controls	Yes / No	*[details]*
File Upload Protection	Yes / No	*[malware scanning approach]*

3.3.4 Environments

[Recommended]

Environment	Description	Count & Venue	Compute Solution
Development	Software development only	*[...]*	*[...]*
Test / QA	Component and integration testing	*[...]*	*[...]*
Staging / Pre-Production	Production-like environment for validation	*[...]*	*[...]*
Production	Live service environment	*[...]*	*[...]*

Environment	Description	Count & Venue	Compute Solution
DR	Disaster recovery environment	*[...]*	*[...]*

Connectivity Between Environments

Does the solution require connectivity between environment tiers (e.g., production to non-production)?

☐ Yes - *[describe which components and data flows]*
☐ No

3.3.5 End User Compute & IoT

[Comprehensive]

End User Compute

Document any end-user device requirements (VDI, BYOD, mobile, desktop software):

[...]

IoT Devices

Document any IoT devices (printers, scanners, cameras, sensors, etc.):

[...]

3.3.6 Sustainability Considerations

[Recommended]

The Physical View is where most carbon-impact decisions are made. Document the sustainability stance for the infrastructure choices above — full detail belongs in Section 4.5, but capture the headline decisions here.

Question	Response
Have hosting regions been chosen for low carbon intensity (e.g., regions with high renewable energy)?	Yes / No — *[which regions and why]*
Are non-production environments configured to auto-shutdown out of hours?	Yes / No — *[schedule]*
Has the compute family been chosen for performance-per-watt (e.g., ARM/Graviton, latest-generation)?	Yes / No — *[details]*
Is auto-scaling configured to release capacity when idle?	Yes / No — *[trigger thresholds]*
Is the DR strategy proportionate (cold standby vs warm vs hot) to the actual recovery objective?	*[describe and rationale]*

Why this matters

Always-on production at peak-sized infrastructure 24×7 is the most common sustainability anti-pattern. Three decisions in this view dominate carbon footprint: **region selection** (carbon intensity varies 5-10× across cloud regions), **non-production auto-shutdown** (typically 60-70% saving on dev/test compute), and **right-sizing** (over-provisioned VMs waste energy regardless of load).

Scoring Guidance

Score	What This Looks Like
1	Hosting venue identified but infrastructure not specified
3	Deployment diagram complete, compute sized, networking documented, environments listed
5	All of the above plus connectivity protocols specified, user/admin access methods documented, security agents listed, bandwidth and latency requirements quantified, sustainability decisions captured

Quality Attribute Cross-References: - **4.2 Reliability** - Infrastructure design directly determines availability and recovery capability - **4.3 Performance** - Compute sizing and network design affect performance - **4.4 Cost** - Infrastructure choices are the primary cost driver - **4.5 Sustainability** - Hosting venue and compute efficiency affect environmental impact

3.4 Data View

[Minimum] `TOGAF Data ISO 42010`

Purpose

The Data View describes how data is stored, classified, protected, and managed throughout its lifecycle. It addresses the concerns of data architects, DBAs, compliance officers, and privacy teams.

3.4.1 Data Architecture & Storage

[Minimum]

Data Footprint

Document all data stores used by the solution, including business data, logs, caches, and temporary stores:

Data Name	Store Technology	Authoritative?	Retention Period	Data Size	Classification	Personal Data?	Encryption Level	Key Management
[name]	*[e.g., PostgreSQL, S3, Redis]*	Yes / No	*[period]*	*[size]*	Public / Internal / Restricted / Highly Restricted	Yes / No	None / Storage / Container / Application	*[method]*

Guidance

For each data store, document: - **Data Store Technology** - The specific technology (e.g., SQL Server, S3 Bucket, Azure Blob) - **Authoritative Data Store** - Is this the master/authoritative store of the data? - **Retention Period** - How long is data retained before deletion? - **Security Classification** - Highest classification of data in the store - **Personal Data** - Does it contain PII or sensitive personal data (SPI)? - **Encryption** - Level of encryption applied (storage, container, application) - **Key Management** - How encryption keys are managed (vendor-managed, HSM, custom)

Storage Systems

[Recommended]

If specific storage systems need provisioning:

Attribute	Detail
Storage Product	*[vendor / product]*
Storage Size	*[TB]*
Storage Type	SAN / NAS / Object / Block / File
Storage Protocol	NFS / SMB / iSCSI / Other
Replication	Synchronous / Asynchronous / Snapshot / None
Minimum RPO	*[recovery point objective]*

3.4.2 Data Classification

[Recommended]

Summarise the data classification profile of the solution:

Classification Level	Data Types	Handling Requirements
Public	*[...]*	Open access
Internal / Corporate	*[...]*	Internal access controls
Restricted	*[...]*	Encrypted, access-controlled, audited
Highly Restricted	*[...]*	Encrypted, strict access, enhanced monitoring

3.4.3 Data Lifecycle

[Recommended]

Describe how data moves through its lifecycle:

Stage	Description	Controls
Creation / Ingestion	How data enters the solution	*[validation, classification]*
Processing	How data is transformed or used	*[access controls, audit]*
Storage	How data is stored at rest	*[encryption, backup]*
Sharing / Transfer	How data is shared or moved	*[encryption in transit, authorisation]*
Archival	How data is archived	*[archive storage, retrieval SLA]*
Deletion / Purging	How data is securely removed	*[secure deletion method, schedule]*

3.4.4 Data Privacy & Protection

[Recommended]

Privacy Assessments

Assessment Type	ID	Status	Link
[PIA / DPIA / LIA]	*[ID]*	*[status]*	*[link]*

Use of Production Data for Testing

Approach	Selected
Only Public/Internal data is used	[ ]
Restricted/Highly Restricted attributes are deleted first	[ ]
Sensitive data is masked (describe method below)	[ ]
Sensitive production data is used (provide justification below)	[ ]
Production data is not used for testing	[ ]

[Additional details on data masking, access controls, duration, and destruction]

Data Integrity

Does the solution include specific data integrity controls beyond standard hardware, protocol, and access controls?

☐ Yes - *[describe integrity controls]*
☐ No

Data on End User Devices

Does the solution allow data to be downloaded to or stored on end-user devices?

☐ Yes - *[describe data protection measures]*
☐ No

3.4.5 Data Transfers & Sovereignty

[Recommended]

Data Transfers to Third Parties

Destination	Data Type	Classification	Transfer Method	Protection
[third party]	*[data description]*	*[classification]*	*[method]*	*[encryption, contracts]*

Data Sovereignty

Are there data sovereignty or residency requirements?

☐ Yes - *[describe requirements and how they are met]*
☐ No

3.4.6 Sustainability Considerations

[Recommended]

Data has a carbon cost — to store, to move, and to back up. Capture how the design minimises that cost. Detail belongs in Section 4.5; decisions belong here.

Question	Response
Are retention periods set to the minimum the regulator and business actually need?	Yes / No — *[per data store]*
Is older data tiered to cold/archive storage (S3 Glacier, Azure Archive, etc.)?	Yes / No — *[tiering policy]*
Are unused or duplicate replicas of data identified and removed?	Yes / No — *[review cadence]*
Is compression applied to reduce storage and transfer volume?	Yes / No — *[formats]*
Is cross-region or cross-cloud replication justified by an actual recovery requirement?	Yes / No — *[justification]*
Are large data transfers scheduled to off-peak / low-carbon-intensity windows where feasible?	Yes / No / Not applicable

Why this matters

Every byte stored consumes electricity for as long as it exists. Two patterns drive most of the waste: **over-retention** (keeping data "just in case" for years past the regulator's requirement) and **uncompressed cold data** (raw logs, raw snapshots, full-fidelity replicas of data that's never read). The fix is policy-led: classify, set retention, automate expiry, tier the rest.

Scoring Guidance

Score	What This Looks Like
1	Data stores listed but classification and retention not defined
3	All data stores classified, retention and encryption specified, PII/SPI identified
5	All of the above plus data sovereignty addressed, data transfers documented with encryption, data integrity controls evidenced, retention/compression aligned to sustainability goals

Quality Attribute Cross-References: - **4.2 Reliability** - Data backup and recovery strategies - **4.4 Cost** - Storage costs, data transfer costs - **4.5 Sustainability** - Data volume minimisation, efficient storage formats

3.5 Security View

[Minimum] `AWS Security` `Azure Security`

Purpose

The Security View describes how the solution protects against threats, manages identity and access, and safeguards data. It addresses the concerns of security architects, the CISO office, compliance teams, and auditors.

:::note[Calibrating effort to risk] This section can be substantial for high-criticality solutions. For **Tier 4 (Low Impact)** or **Tier 5 (Minimal)** solutions, complete only the Minimum-tagged subsections: Business Impact Assessment, basic Authentication, and Encryption at Rest. The Threat Model, Session Management details, and Privileged Access can be skipped for low-criticality projects unless your governance specifically requires them. See the Depth Cheat Sheet for what's required at each depth. :::

3.5.1 Security Overview & Threat Model

[Minimum]

Security Context

Question	Response
Does the solution support regulated activities?	Yes / No - *[details]*
Is the solution SaaS or third-party hosted?	Yes / No - *[details]*
Has a third-party risk assessment been completed?	Yes / No - *[reference]*

Business Impact Assessment

Impact Category	Business Impact if Compromised
Confidentiality	*[impact of unauthorised data access]*
Integrity	*[impact of unauthorised data modification]*
Availability	*[impact of service unavailability]*
Non-Repudiation	*[impact if actions can be denied]*

Guidance

Rate each category as Critical / High / Medium / Low / Negligible. The highest rating drives the security requirements for the solution — a system with Critical confidentiality needs stronger encryption and access controls than one rated Low. Use your organisation's data classification policy to inform these ratings.

Threat Model

[Comprehensive]

Document the key threats and how the design addresses them:

Threat	Attack Vector	Likelihood	Impact	Mitigation
[threat]	*[how]*	H / M / L	H / M / L	*[design controls]*

3.5.2 Identity & Access Management

[Minimum]

Authentication Model

[Minimum]

Internal Users

Access Type	Role(s)	Destination(s)	Authentication Method	Credential Protection
End Users	*[roles]*	*[servers/apps]*	*[SSO/SAML/OIDC/etc.]*	*[method]*
IT Operations	*[roles]*	*[servers/apps]*	*[method]*	*[method]*
Service Accounts	*scope*	*[servers/apps]*	*[method]*	*[method]*

External Users (if applicable)

Access Type	Role(s)	Destination(s)	Authentication Method	Credential Protection
External Users	*[roles]*	*[servers/apps]*	*[method]*	*[method]*
External Applications	*scope*	*[servers/apps]*	*[method]*	*[method]*

Guidance

Document every authentication path into the solution. Prefer federated identity (SSO, OIDC, SAML — see glossary for definitions) over local credentials. Consider: - **Internal users** — should use the organisation's identity provider (e.g., Entra ID, Okta) - **External users** — may need a separate identity provider or customer-facing auth (e.g., OAuth 2.0) - **Service accounts** — prefer managed identity or workload identity over shared secrets - **API consumers** — document whether mutual TLS (mTLS), API keys, or OAuth client credentials are used

Authentication Details

For each authentication method identified above, document the detailed configuration:

[Recommended]

Control	Response
Does the application use SSO or group-wide authentication?	*[...]*
What is the unique identifier for user accounts?	*[...]*
What is the authentication flow?	*[describe or diagram]*
How are credentials issued to users?	*[...]*
What are the credential complexity rules?	*[...]*
What are the credential rotation rules?	*[...]*
What are the account lockout rules?	*[...]*
How can users reset forgotten credentials?	*[...]*

Session Management

Session management controls prevent session hijacking and ensure users are re-authenticated appropriately.

Document how authenticated sessions are maintained and protected:

[Recommended]

Control	Response
How are sessions established after authentication?	*[tokens, cookies, etc.]*
How are session tokens protected against misuse?	*[signing, encryption, etc.]*
What are the session timeout and concurrency limits?	*[...]*

Authorisation Model

Document how access rights are assigned and controlled after authentication:

[Minimum]

Access Type	Role / Scope	Entitlement Store	Provisioning Process
Business Users	*[roles and scope]*	*[where entitlements stored]*	*[how provisioned]*
Technology Users	*[roles and scope]*	*[where entitlements stored]*	*[how provisioned]*
Service Accounts	*[roles and scope]*	*[where entitlements stored]*	*[how provisioned]*

Authorisation Details

[Recommended]

Control	Response
Account re-certification process	*[...]*
Segregation of duties controls	*[...]*
Delegated authorisation capabilities	*[...]*

Privileged Access

[Recommended]

Account Type	Management Approach
OS privileged accounts (root/admin)	*[how managed, rotated, audited]*
Infrastructure / platform admin	*[how managed]*
Application admin	*[how managed]*

Guidance

Privileged access is a primary attack vector. Document how privileged accounts are: - **Controlled** — who can request access, what approval is needed - **Time-limited** — prefer just-in-time (JIT) access over standing privileges - **Audited** — session recording, command logging, periodic review - **Break-glass** — emergency access process with post-incident review

3.5.3 Network Security & Perimeter Protection

[Recommended]

Describe network segmentation, firewalls, and perimeter controls. Cross-reference Section 3.3.3 (Network Topology) for the underlying infrastructure.

Control	Implementation
Network segmentation	*[firewalls, security groups, NSGs]*
Ingress filtering	*[WAF, DDoS, rate limiting]*
Egress filtering	*[proxy, gateway, allowlists]*
Encryption in transit	*[TLS versions, certificate management]*

3.5.4 Data Protection

[Recommended]

Cross-reference Section 3.4 (Data View) for the full data architecture. Document the security controls applied to the solution.

Encryption at rest and secret management protect sensitive data from exposure if storage or infrastructure is compromised.

Encryption at Rest

Encryption protects data from unauthorised access even if storage media is compromised.

Attribute	Detail
Encryption deployment level	Storage / Logical Container / Application
Key type	Symmetric / Asymmetric
Algorithm / cipher / key length	*[…]*
Key generation method	*[HSM / software / other]*
Key storage	*[HSM / KMS / other]*
Key rotation schedule	*[…]*

Secret & Password Protection

Attribute	Detail
Secret store	*[e.g., HashiCorp Vault, AWS Secrets Manager, Azure Key Vault]*
Secret distribution	Distributed at deployment / Retrieved on-demand
Secret protection on host	Local vault / File system / Memory only / Not stored
Secret rotation	Automatic / Manual / Not rotated

3.5.5 Security Monitoring & Threat Detection

[Recommended]

Capability	Implementation
Security event logging	*[what events, where stored]*
SIEM integration	*[tool and integration method]*
Infrastructure event detection	*[how monitored]*
Security alerting	*[how alerts are generated and routed]*

[Insert security monitoring architecture diagram if applicable]

Scoring Guidance

Score	What This Looks Like
1	Authentication method identified but controls not detailed
3	Business impact assessed, authentication and authorisation documented, encryption at rest and in transit specified
5	All of the above plus threat model complete, secrets management documented, security monitoring integrated with SIEM, privileged access controlled with JIT

Quality Attribute Cross-References: - **4.1 Operational Excellence** - Security monitoring is part of operational observability - **4.2 Reliability** - Security incidents affect availability

3.6 Scenarios

[Recommended] `4+1 Scenarios` `ISO 42010`

Purpose

Scenarios are the "+1" in the 4+1 View Model. They serve as the connective tissue between all other views, validating that the architecture works end-to-end for key use cases. In ISO 42010 terms, they document **correspondences** between views.

This section also captures Architecture Decision Records (ADRs) - the significant decisions that shaped the design and the rationale behind them.

3.6.1 Key Use Cases

[Recommended]

Document the most architecturally significant use cases. Each use case should trace through multiple views to validate the design:

Use Case Template

UC-[nn]: [Use Case Name]

Attribute	Detail
Actor(s)	*[who initiates this use case]*
Trigger	*[what starts the flow]*
Pre-conditions	*[required state before execution]*
Main Flow	*[step-by-step description]*
Post-conditions	*[state after successful execution]*
Views Involved	Logical / Integration & Data Flow / Physical / Data / Security

Guidance

Select 3-5 use cases that are most architecturally significant: - The primary happy-path user journey - A high-volume or performance-critical flow - A failure/recovery scenario - A security-relevant scenario (e.g., authentication, data access) - An integration-heavy scenario

Each use case should demonstrate how components from the Logical View communicate (Integration & Data Flow View), are deployed (Physical View), handle data (Data View), and are secured (Security View).

3.6.2 Architecture Decision Records (ADRs)

[Recommended]

ADRs live in this section because they emerge from working through the architectural views and use cases above. As you describe how the solution behaves, you naturally encounter trade-offs and decisions worth recording. A summary index of all ADRs also appears in Section 6.8 for governance review.

Document the key architectural decisions using the ADR format:

ADR Template

ADR-[nnn]: [Decision Title]

Field	Content
Status	Proposed / Accepted / Superseded / Deprecated
Date	*[decision date]*
Context	*[what is the issue or question]*
Decision	*[what was decided]*
Alternatives Considered	*[what other options were evaluated]*
Consequences	*[positive and negative implications]*
Quality Attribute Tradeoffs	*[which quality attributes are affected and how]*

Guidance

Record decisions that: - Are difficult or expensive to reverse - Involve significant tradeoffs between quality attributes - Deviate from standard patterns or policies - Were debated or where multiple valid approaches existed - Affect multiple views or stakeholders

ADRs should be immutable once accepted. If a decision is reversed, create a new ADR that supersedes the original rather than editing it.

Example ADRs to Consider

Category	Example Decision
Platform	Choice of cloud provider, hosting venue, service model
Architecture	Monolith vs. microservices, event-driven vs. request-response
Data	Choice of database technology, caching strategy
Security	Authentication approach, encryption standards
Integration	Synchronous vs. asynchronous, API style
Resilience	Active-active vs. active-passive, DR strategy

Scoring Guidance

Score	What This Looks Like
1	One or two use cases sketched; no ADRs
3	Key use cases documented with actors and flows; significant decisions captured as ADRs with rationale

Score	What This Looks Like
5	All of the above plus use cases cross-reference all relevant views; ADRs include alternatives considered and quality attribute impact

4. Architectural Quality Attributes

`AWS WAF` `Azure WAF` `GCP WAF`

Purpose

Architectural Quality Attributes are **cross-cutting quality checks — evaluated across all views rather than within a single view**. They are derived from the Well-Architected Frameworks published by AWS, Azure, GCP, Oracle, and IBM, unified into quality attributes that address the key architectural concerns of any solution.

Unlike the Architectural Views (Section 3), which describe **what** the solution is, Quality Attributes evaluate **how well** it is designed.

Quality Attributes

#	Quality Attribute	Focus
4.1	**Operational Excellence**	Observability, monitoring, operational procedures
4.2	**Reliability & Resilience**	DR, scalability, fault tolerance, backup/recovery
4.3	**Performance Efficiency**	Performance requirements, resource optimisation
4.4	**Cost Optimisation**	Cost analysis, FinOps, cost-effective design
4.5	**Sustainability**	Energy, carbon, resource efficiency

These quality attributes are derived from the Well-Architected Frameworks published by AWS, Azure, GCP, Oracle, and IBM. For the full provider-by-provider mapping, see the Framework Alignment page.

:::note[Security as a View, Not Just a Quality Attribute] In the cloud Well-Architected Frameworks, **Security** is a quality attribute (pillar). In this standard, Security is elevated to a full **Architectural View** (Section 3.5) because of the depth of documentation required for IAM, encryption, network security, and monitoring. The Security View directly covers the security concerns that cloud

providers treat as a quality pillar — so there is no separate security score in the quality attributes table. :::

How to Use Quality Attributes

1. **Design Phase** — Use each quality attribute's guidance to inform design decisions
2. **Review Phase** — Evaluate the completed design against each quality attribute
3. **Tradeoff Documentation** — Where quality attribute requirements conflict (e.g., cost vs. resilience), document the tradeoff and the rationale for the chosen balance
4. **Cross-Reference** — Reference specific quality attribute concerns within the Architectural Views using the quality attribute callout format (the coloured cross-reference boxes at the bottom of each view page)

Quality Attribute Tradeoffs

Optimising for one quality attribute may come at the expense of another. Key tradeoffs to consider:

Tradeoff	Description
Reliability vs. Cost	Higher availability requires redundant infrastructure
Performance vs. Cost	Better performance may require more powerful (expensive) resources
Security vs. Usability	Stronger security controls may increase friction for users
Sustainability vs. Performance	Energy-efficient choices may not be the highest-performing
Reliability vs. Sustainability	DR environments and redundancy consume additional resources

Document any significant tradeoffs in Section 3.6 (Scenarios / ADRs).

4.1 Operational Excellence

[Recommended] `AWS Ops Excellence` `Azure Ops Excellence` `GCP Ops Excellence`

Purpose

Operational Excellence focuses on how the solution is monitored, operated, and improved over time. It covers observability (logging, metrics, tracing), alerting, capacity management, and operational procedures. Evaluate this quality attribute across all architectural views documented in Section 3.

4.1.1 Observability - Logging

[Recommended]

Log Architecture

Log Type	Events Logged	Local Storage	Retention Period	Remote Services
Application logs	*[what is logged]*	*[file system, database]*	*[period]*	*[e.g., Datadog, CloudWatch]*
Data store logs	*[what is logged]*	*[location]*	*[period]*	*[remote service]*
Infrastructure logs	*[what is logged]*	*[location]*	*[period]*	*[remote service]*
Security event logs	*[what is logged]*	*[location]*	*[period]*	*[SIEM service]*

Guidance

For each log type, document: - What events are captured (application errors, access logs, audit events, etc.) - Where logs are stored locally within the application - How long logs are retained before rotation or deletion - Whether logs are forwarded to centralised logging or SIEM services

4.1.2 Observability - Monitoring & Alerting

[Recommended]

Operational Alerts

Describe how operational alerts are implemented:

Alert Category	Trigger Condition	Notification Method	Recipient
[e.g., Application error rate]	*[threshold]*	*[email, PagerDuty, Slack]*	*[team/role]*

Monitoring Tools

Capability	Tool	Coverage
Application Performance Monitoring	*[e.g., Datadog, New Relic]*	*[which components]*
Infrastructure Monitoring	*[e.g., CloudWatch, Prometheus]*	*[which resources]*
Log Aggregation	*[e.g., ELK, Splunk, Datadog Logs]*	*[which log sources]*
Distributed Tracing	*[e.g., Jaeger, X-Ray, Datadog APM]*	*[which services]*

4.1.3 Capacity Monitoring

[Comprehensive]

Question	Response
What metrics are collected for capacity monitoring?	*[CPU, memory, storage, network, queue depth]*
How are capacity trends analysed?	*[tools, dashboards, reports]*
Are capacity thresholds and alerts configured?	*[threshold details]*
Is there a capacity planning process?	*[process description]*

4.1.4 Operational Procedures

[Comprehensive]

Document key operational procedures and runbooks:

Procedure	Description	Owner	Documentation
Incident response	*[how incidents are detected and resolved]*	*[team]*	*[link]*
Change management	*[how changes are approved and deployed]*	*[team]*	*[link]*
Escalation paths	*[escalation procedures]*	*[team]*	*[link]*
On-call rotation	*[on-call structure]*	*[team]*	*[link]*

Scoring Guidance

Score	What This Looks Like
1	Monitoring tool identified but not configured
3	Centralised logging, monitoring, and alerting in place; runbooks documented
5	All of the above plus distributed tracing enabled, dashboards defined, incident response procedures tested

4.2 Reliability & Resilience

[Recommended] `AWS Reliability` `Azure Reliability` `GCP Reliability`

Purpose

Reliability & Resilience focuses on the solution's ability to withstand failures, recover from disruptions, and scale to meet demand. This quality attribute is closely tied to the Physical View (3.3) and Data View (3.4) for infrastructure and data backup details. Evaluate this quality attribute across all architectural views documented in Section 3.

4.2.1 Geographic Footprint & Disaster Recovery

[Recommended]

Question	Response
Is the application deployed across multiple hosting venues for continuity?	Yes / No - *[details]*
What is the DR strategy?	Active-Active / Active-Passive / Pilot Light / Backup & Restore
Are there data sovereignty requirements affecting geographic choices?	Yes / No - *[details]*

[Insert geographic deployment diagram if applicable]

4.2.2 Scalability

[Recommended]

Application Scalability

Attribute	Response
Scaling capability	No dynamic scaling (pre-sized) / Manual scaling / Partial auto-scaling / Full auto-scaling
Scaling details	*[describe how scaling works, triggers, limits]*

Dependency Scalability

Attribute	Response
Dependencies adequately sized?	Yes (confirmed) / Unconfirmed / Known insufficient
Dependency details	*[describe scaling posture of dependencies]*

4.2.3 Fault Tolerance

[Recommended]

Has the application been designed to tolerate unexpected disruptions such as failure or degradation of internal components or external dependencies?

- ☐ **Yes** - *[describe fault tolerance design, including:]*
  - How the application handles component failures
  - Graceful degradation strategies
  - Circuit breaker or retry patterns
  - Health check and self-healing mechanisms
  - Testing practices (chaos engineering, game days)
- ☐ **No** - *[describe why not]*

4.2.4 Failure Modes & Recovery Behaviour

[Recommended]

Document how the solution behaves when individual components or dependencies fail:

Component / Dependency	Failure Mode	Detection Method	Recovery Behaviour	User Impact
[component]	*[how it fails]*	*[how detected: health check, alert, timeout]*	*[auto-restart, failover, graceful degradation, manual intervention]*	*[full outage, degraded service, transparent]*

Guidance

For each critical component, consider: - **What happens when it becomes unavailable?** Does the solution fail entirely, degrade gracefully, or continue with reduced functionality? - **How is the failure detected?** Health checks, heartbeats, error thresholds, timeouts? - **How is it recovered?** Automatic restart, failover to secondary, circuit breaker, manual intervention? - **What do users experience?** Full outage, degraded experience, increased latency, or transparent failover?

This section is frequently missing from architecture documents but is one of the most valuable for operations teams and SREs.

4.2.5 Backup & Recovery

[Recommended]

Backup Design

Attribute	Detail
Backup strategy	*[what is backed up and how]*
Backup product/service	*[tool used]*
Backup type	Full / Incremental / Differential
Backup frequency	*[schedule]*
Backup retention	*[period]*

Backup Protection

Control	Detail
Immutability	*[how backups are protected against modification/deletion]*
Encryption	*[how backup data is encrypted]*
Access control	*[who can access backups]*

4.2.6 Recovery Scenarios

[Recommended]

Document how the solution recovers under different failure scenarios:

#	Scenario	Recovery Approach	RTO	RPO
1	Primary hosting venue / AZ / region failure	*[approach]*	*[time]*	*[time]*
2	Critical software component failure	*[approach]*	*[time]*	*[time]*
3	Key infrastructure failure (hardware, storage, network)	*[approach]*	*[time]*	*[time]*
4	Network connectivity failure between venues	*[approach]*	*[time]*	*[time]*
5	External connectivity failure (customer-facing)	*[approach]*	*[time]*	*[time]*
6	Ransomware / cyber-attack	*[approach]*	*[time]*	*[time]*
7	Accidental or malicious data corruption / deletion	*[approach]*	*[time]*	*[time]*

Guidance

For each scenario, describe: - How the failure is detected - Automatic vs. manual recovery steps - Expected Recovery Time Objective (RTO) and Recovery Point Objective (RPO) - Any dependencies on other teams or systems for recovery - Whether recovery has been tested and when

Scoring Guidance

Score	What This Looks Like
1	DR strategy identified but RTO/RPO not defined
3	DR strategy documented with RTO/RPO targets, backup configured, scalability approach defined
5	All of the above plus fault tolerance designed, chaos testing practised, backup immutability and encryption confirmed, DR tested

4.3 Performance Efficiency

[Recommended] `AWS Performance` `Azure Performance` `GCP Performance`

Purpose

Performance Efficiency focuses on using computing resources efficiently to meet requirements and maintaining that efficiency as demand changes and technologies evolve. Evaluate this quality attribute across all architectural views documented in Section 3.

4.3.1 Performance Requirements

[Recommended]

Key Performance Indicators

Metric	Target	Measurement Method
Response time (P50 / P95 / P99)	*[e.g., < 200ms / < 500ms / < 1s]*	*[how measured]*
Throughput	*[e.g., 1000 requests/sec]*	*[how measured]*
Concurrent users	*[e.g., 10,000 simultaneous]*	*[how measured]*
Batch processing time	*[e.g., < 2 hours for daily batch]*	*[how measured]*
Data processing latency	*[e.g., < 5 seconds end-to-end]*	*[how measured]*

Guidance

Set targets based on real user expectations, not arbitrary numbers. Consider: - **P95/P99 response times** — these matter more than averages; a 200ms average with a 5s P99 is a poor experience - **Throughput** — derive from expected user volumes and usage patterns, not theoretical maximums - **Batch processing** — define the acceptable window (e.g., must complete before business hours) - **Measurement method** — specify whether targets are measured at the client, load balancer, or application layer

Performance Testing

Attribute	Detail
Performance testing approach	*[load testing, stress testing, soak testing]*
Testing tools	*[e.g., JMeter, Gatling, k6, Locust]*
Testing environment	*[which environment, how it compares to production]*
Testing frequency	*[when performance tests are run]*

Capacity & Growth Projections

[Recommended]

Metric	Current	1 Year	3 Years	5 Years
Users (total)	*[n]*	*[n]*	*[n]*	*[n]*
Concurrent users (peak)	*[n]*	*[n]*	*[n]*	*[n]*
Data volume	*[n GB/TB]*	*[n GB/TB]*	*[n GB/TB]*	*[n GB/TB]*
Transaction volume (per day)	*[n]*	*[n]*	*[n]*	*[n]*
Storage requirement	*[n GB/TB]*	*[n GB/TB]*	*[n GB/TB]*	*[n GB/TB]*

Question	Response
Will the current design scale to accommodate projected growth?	Yes / No — *[if no, describe what changes will be needed]*
Are there known seasonal or cyclical demand patterns?	Yes / No — *[if yes, describe peak periods]*

4.3.2 Resource Optimisation

[Comprehensive]

Document how the solution optimises resource usage:

Strategy	Implementation
Right-sizing	*[how compute resources are sized appropriately]*
Caching	*[caching layers and strategies]*
Connection pooling	*[database and service connection management]*
Asynchronous processing	*[use of queues, background jobs]*
Content delivery	*[CDN usage for static assets]*
Database optimisation	*[indexing strategy, query optimisation]*

4.3.3 Network Performance

[Comprehensive]

Cross-reference Section 3.3.3 for network topology. This section documents performance-specific network considerations:

Attribute	Detail
Latency requirements	*[round-trip time expectations]*
Bandwidth requirements	*[data transfer volumes]*
QoS requirements	*[quality of service needs]*
Content delivery strategy	*[CDN, edge caching, regional deployment]*
Network optimisation	*[compression, connection reuse, protocol choice]*

Scoring Guidance

Score	What This Looks Like
1	General performance expectations stated but no targets
3	Response time, throughput, and concurrency targets defined; growth projections documented
5	All of the above plus performance testing automated, caching strategy documented, resource optimisation evidenced, seasonal patterns accounted for

4.4 Cost Optimisation

[Recommended] `AWS Cost` `Azure Cost` `GCP Cost`

Purpose

Cost Optimisation focuses on delivering the solution at the lowest reasonable cost while meeting all other quality requirements. It covers cost-aware design decisions, cost analysis, and ongoing cost management practices. Evaluate this quality attribute across all architectural views documented in Section 3.

4.4.1 Cost Influence & Analysis

[Recommended]

Design Cost Decisions

How do the design decisions in hosting, compute, networking, data storage, and resiliency demonstrate cost-effectiveness?

Posture	Selected	Detail
Design decisions chosen for specific reasons other than cost	[ ]	*[describe why]*
Most cost-effective options intentionally not selected	[ ]	*[describe why and the tradeoffs]*
Most cost-effective options selected	[ ]	*[describe the analysis]*

Cost Analysis

Has cost analysis or modelling been performed to inform design decisions?

- ☐ **No** - A cost analysis has not been performed
- ☐ **Yes** - *[describe the analysis approach, tools used (e.g., cloud pricing calculators, TCO models), and key findings]*

4.4.2 Cost Implications

[Recommended]

Has the design been constrained in ways that do not fully meet requirements because of cost?

☐ **No** - The design fully meets requirements; cost has not impacted the design
☐ **Yes** - *[describe how and why cost constrained the design, and the impact]*

4.4.3 FinOps Practices

[Comprehensive]

Document ongoing cost management practices:

Practice	Implementation
Cost monitoring	*[dashboards, alerts, tools]*
Cost allocation	*[tagging strategy, cost centres]*
Reserved capacity	*[reserved instances, savings plans, committed use]*
Rightsizing reviews	*[process and frequency for reviewing resource utilisation]*
Waste elimination	*[process for identifying and removing unused resources]*
Budget governance	*[budget alerts, approval thresholds]*

Guidance

Consider documenting: - Expected monthly/annual run cost breakdown by component - Cost comparison of alternatives considered - Cost optimisation opportunities identified but deferred - Triggers that would necessitate cost re-evaluation (e.g., scale events, contract renewals)

Scoring Guidance

Score	What This Looks Like
1	Cost acknowledged as a concern but no analysis performed
3	Cost analysis completed, monitoring enabled, tagging strategy in place
5	All of the above plus reserved capacity assessed, right-sizing evidenced, cost model linked to cloud calculator, FinOps practices documented

4.5 Sustainability

[Recommended] `AWS Sustainability` `Azure Sustainability` `GCP Sustainability`

Purpose

Sustainability focuses on minimising the environmental impact of the solution by optimising the use of compute, storage, and network resources. This quality attribute was introduced by the AWS Well-Architected Framework and addresses energy consumption, carbon emissions, and resource efficiency. Evaluate this quality attribute across all architectural views documented in Section 3.

Where the architect's decisions actually live

Sustainability is decided at the point of architecture choice, not retrofitted at scoring time. Concrete sustainability prompts are embedded in the views and lifecycle so they are confronted when the decision is being made, not parked in a separate section that gets filled in last:

Where	What's captured
3.1.7 Logical View — Sustainability Considerations	Caching, async patterns, batch consolidation, framework weight
3.3.6 Physical View — Sustainability Considerations	Region carbon intensity, auto-shutdown, ARM/efficient compute, auto-scaling, DR posture
3.4.6 Data View — Sustainability Considerations	Retention minimisation, tiering, replication justification, compression
5.5 Operations & Support — Sustainability in Operation	Auto-shutdown schedules, right-sizing cadence, orphan-resource reclamation, retirement
Cheat Card 6 — Sustainability Quick Hits	One-page summary of where carbon footprint is actually decided

The remainder of this section captures the underlying measurement and reporting (carbon baseline, SCI metric, code/network/data efficiency) that supports those view-level decisions.

Questions the architect should be able to answer

Before signing off the SAD, the architect should be able to answer these without consulting the team:

1. **Region choice** — *why this region for this workload, and what's its carbon intensity relative to alternatives?*
2. **DR proportionality** — *is the DR posture (cold / pilot light / warm / hot) matched to the actual RTO, or is it always-on by default?*
3. **Non-production discipline** — *what's the auto-shutdown schedule for dev/test, and how is it enforced?*
4. **Data retention** — *what's the retention period for each data store, and is it the minimum the regulator and business need?*
5. **The biggest contributor** — *what's the largest single source of this solution's carbon footprint, and what would it take to halve it?*

If any of these has no answer, the design is not yet sustainability-complete at Recommended depth.

4.5.1 Hosting Efficiency

[Recommended]

Hosting Location

Question	Response
Has the hosting location been chosen to reduce environmental impact?	Yes / No - *[describe criteria: CO2 intensity, renewable energy, water usage]*
What is the expected workload demand pattern?	Variable / Constant - *[describe peaks and patterns]*

On-Demand Availability

Question	Response
Must the application be available continuously?	Yes / No
Can the solution be shut down or scaled down during off-peak hours?	Yes / No - *[describe schedule]*
Are non-production environments configured to downscale or shut down when not in use?	Yes / No - *[details]*

Resource Efficiency

Question	Response
Are resources rightsized to avoid overprovisioning?	*[describe approach]*
Is vCPU utilisation monitored?	*[target utilisation %]*

Question	Response
Are the highest performance-per-watt hardware options used?	*[details]*

4.5.2 Carbon Measurement

[Recommended]

Question	Response
Has a carbon footprint baseline been established for the solution?	Yes / No
Which cloud provider carbon reporting tool is used?	AWS Customer Carbon Footprint Tool / Azure Emissions Impact Dashboard / Google Carbon Footprint / Other / None / N/A
Is there a target for carbon reduction?	Yes / No — *[target and timeframe]*
Is the Software Carbon Intensity (SCI) metric tracked?	Yes / No

Guidance

The Green Software Foundation defines three principles for sustainable software: - **Carbon efficiency** — emit the least carbon possible - **Energy efficiency** — use the least energy possible - **Carbon awareness** — do more when the electricity is cleaner, less when it is dirtier

The **Software Carbon Intensity (SCI)** specification provides a standard metric for measuring the carbon impact of software: `SCI = ((E * I) + M) per R` where E = energy, I = carbon intensity of electricity, M = embodied emissions, R = functional unit. See sci.greensoftware.foundation.

4.5.3 Network Efficiency

[Comprehensive]

Question	Response
Is a CDN used to reduce origin server traffic?	Yes / No
Is edge computing used to process data closer to users?	Yes / No
Are efficient protocols used (e.g., HTTP/3, gRPC, protocol buffers)?	Yes / No — *[details]*
Is cross-region data transfer minimised?	Yes / No — *[details]*
Are data compression strategies applied to reduce transfer volumes?	Yes / No

4.5.4 Code Efficiency

[Comprehensive]

Question	Response
How do the language and framework choices contribute to efficiency?	*[describe]*
Has the code been optimised for the target platform and workload?	*[describe]*
Are efficient algorithms and data structures used?	*[describe]*
Is the number of vCPU hours per job/request minimised?	*[describe]*

4.5.5 Data Efficiency

[Comprehensive]

Question	Response
Is data held close to compute to reduce network transfer?	*[details]*
Are data replicas minimised?	*[details]*
Is old or unused data removed to reduce storage?	*[details]*
Are efficient data formats and compression used?	*[details]*
Are jobs prioritised and distributed to optimise resource usage?	*[details]*
Are efficient networking patterns used (reduced distance, data transfer volume)?	*[details]*

Guidance

Consider the environmental impact of: - **DR environments** - Optimal patterns for standby compute (e.g., pilot light vs. warm standby) - **Non-production environments** - Auto-shutdown schedules, right-sized environments - **Data lifecycle** - Tiered storage, data expiry, archive strategies - **Network design** - Minimising cross-region traffic, edge computing

Scoring Guidance

Score	What This Looks Like
1	Sustainability acknowledged but no specific actions taken
3	Hosting location considered for carbon impact, non-production auto-shutdown enabled, resources right-sized, carbon baseline established
5	All of the above plus SCI metric tracked, workload patterns optimised, network efficiency addressed, continuous availability justified where required

5. Lifecycle Management

[Recommended] `4+1 Development` `AWS Ops Excellence` `Azure Ops Excellence`

Purpose

Lifecycle Management describes how the solution is developed, deployed, operated, and eventually retired. It corresponds to the Development View in the 4+1 model and the operational aspects of the AWS/Azure Well-Architected Frameworks.

Sub-section	Focus	Depth
5.1 Software Development & CI/CD	Build and deploy pipelines	**[Recommended]**
5.2 Service Transition & Migration	Migration strategy and cutover	**[Recommended]**
5.3 Test Strategy	Architecturally significant testing	**[Recommended]**
5.4 Release Management	Release frequency and process	**[Recommended]**
5.5 Operations & Support	Support model and SLAs	**[Recommended]**
5.6 Resourcing & Skills	Team capability and readiness	**[Recommended]**
5.7 Service Start	Start-up sequence and dependencies	**[Comprehensive]**
5.8 Maintainability	Patching, certificates, dependencies	**[Recommended]**
5.9 Decommissioning & Legacy Removal	End-of-life and disposal	**[Recommended]**
5.10 Exit Planning	Vendor lock-in and exit strategy	**[Recommended]**

5.1 Software Development & CI/CD

[Recommended]

Development Practices

Does the application include any software developed internally?

- ☐ Yes - *[complete the sections below]*
- ☐ No - *[commercial/vendor solution only]*

Attribute	Detail
Source control platform	*[e.g., GitHub, GitLab, Bitbucket]*
CI/CD platform	*[e.g., Jenkins, GitHub Actions, Azure DevOps]*
Build automation	*[how builds are triggered and managed]*
Deployment automation	*[how deployments are automated]*
Test automation	*[what testing is automated in the pipeline]*

Application Security in Development

Control	Implementation
Security requirements identification	*[how security requirements are captured]*
Static Application Security Testing (SAST)	*[tool used]*
Dynamic Application Security Testing (DAST)	Yes / No - *[tool]*
Software Composition Analysis (SCA)	*[tool used]*
Container image scanning	*[if applicable, tool used]*
Secure coding practices	*[standards, training, code review]*
Patch management	*[how security patches are applied and SLAs]*

5.2 Service Transition & Migration

[Recommended]

Migration Classification (6 R's)

If this solution replaces or migrates an existing system, classify the migration approach:

Classification	Selected?	Description
Retain	☐	Keep as-is, not suitable for migration at this time
Retire	☐	Decommission; functionality no longer needed
Rehost	☐	Lift-and-shift to new infrastructure with minimal changes
Replatform	☐	Lift-and-shift with targeted optimisations (e.g., managed database)
Refactor	☐	Re-architect components to take advantage of new platform capabilities
Replace	☐	Replace entirely with a new solution (e.g., SaaS product)

Transition Plan

Attribute	Detail
Deployment strategy	Big Bang / Blue-Green / Canary / Strangler Fig / Rolling / Parallel Run
Data migration mode	One-off / Phased / Continuous Sync / Not applicable
Data migration method	*[e.g., Export/Import, ETL, CDC, DMS, manual]*
Data volume to migrate	*[e.g., 500 GB]*
End-user cutover approach	One-off / Phased / Not applicable
External system cutover	One-off / Phased / Not applicable
Maximum acceptable downtime	Zero / Seconds / Minutes / Hours / Days
Rollback plan	*[how to revert if the transition fails]*
Acceptance criteria	*[what must be true before cutover]*
Transient infrastructure needed?	Yes / No — *[if yes, describe temporary infrastructure required during migration]*

5.3 Test Strategy

[Recommended]

Describe the testing approach that validates the architecture:

Test Type	Scope	Approach	Environment	Automated?
Integration testing	*[what is tested]*	*[approach]*	*[environment]*	Yes / No
Contract testing	*[API contracts between services]*	*[approach]*	*[environment]*	Yes / No
Performance testing	*[load, stress, soak]*	*[approach]*	*[environment]*	Yes / No
Security testing	*[penetration, vulnerability]*	*[approach]*	*[environment]*	Yes / No
DR testing	*[failover, recovery]*	*[approach and frequency]*	*[environment]*	Yes / No

Guidance

This section covers architecturally significant testing — not unit tests or functional test cases (which belong in detailed design / low-level documentation). Focus on testing that validates the architecture itself: integration points, performance characteristics, security posture, and recovery capabilities.

5.4 Release Management

[Recommended]

Attribute	Detail
Release frequency	*[e.g., continuous, weekly, monthly, quarterly]*
Release process	*[how releases are planned, approved, and deployed]*
Release validation	*[how releases are validated in production]*
Feature flags / toggles	*[if used, how they are managed]*

5.5 Operations & Support

[Recommended]

Attribute	Detail
Support model	*[which teams or vendors support the application]*
Support hours	*[e.g., 24/7, business hours, follow-the-sun]*
SLAs	*[internal or external service level agreements]*
Escalation paths	*[how issues are escalated]*

Sustainability in Operation

[Recommended]

Operational practices have a continuous carbon impact. Document how sustainability is preserved over the life of the running solution; the metric and tooling detail belongs in Section 4.5.

Question	Response
Are non-production environments on an auto-shutdown schedule (e.g., evenings, weekends)?	Yes / No — *[schedule]*
Is there a periodic review of compute right-sizing (typically quarterly)?	Yes / No — *[cadence]*
Are unused resources (orphaned VMs, unattached disks, idle environments) actively identified and reclaimed?	Yes / No — *[process]*
Is the carbon footprint reported alongside cost in operational reviews?	Yes / No
Are environment retirements (e.g., a decommissioned dev cluster) actually deleted, not just stopped?	Yes / No

5.6 Resourcing & Skills

[Recommended]

Assess whether the team has the skills and resources to build, operate, and support the solution.

Team Capability Assessment

Skill Area	Current Level	Action Required
Cloud platform (e.g., AWS, Azure, GCP)	High / Medium / Low / N/A	*[training, hiring, or contractor needed?]*
Infrastructure as Code (e.g., Terraform, Pulumi)	High / Medium / Low / N/A	*[…]*
CI/CD pipeline management	High / Medium / Low / N/A	*[…]*
Application technology stack	High / Medium / Low / N/A	*[…]*
Database administration	High / Medium / Low / N/A	*[…]*
Security & compliance	High / Medium / Low / N/A	*[…]*

Operational Readiness

Question	Response
Can the team fully operate and support this solution in production?	A: Fully capable / B: Partially capable / C: Not yet capable but willing to learn / D: Not capable and will not be in the foreseeable future
If B, C, or D: what additional resources are required?	*[e.g., DBA, DevOps engineer, cloud architect, managed service]*
Is a managed service being considered for ongoing operations?	Yes / No — *[details]*

5.7 Service Start

[Comprehensive]

Describe how the application and its supporting services are started:

[What steps are needed to bring the solution into operation? Are there manual steps? What is the start-up sequence and dependency order?]

5.8 Maintainability

[Recommended]

Describe how the solution design enables ongoing maintenance:

Concern	Approach
Keeping software versions current and supported	*[patching strategy]*

Concern	Approach
Hardware lifecycle management	*[refresh cadence]*
Certificate management	*[renewal process]*
Dependency management	*[how third-party dependencies are tracked and updated]*

5.9 Decommissioning & Legacy Removal

[Recommended]

Where this solution replaces or modifies existing systems, document the plan for removing legacy infrastructure.

Solution Lifespan

Attribute	Detail
Intended lifespan	*[expected or planned lifetime of the solution]*
End-of-life triggers	*[what would trigger decommissioning]*
Decommissioning blockers	*[dependencies or constraints on retirement]*

Legacy Infrastructure Removal

Legacy Component	Current State	Decommission Date	Owner	Dependencies	Status
[system/server/service]	*[running / standby / powered off]*	*[target date]*	*[person/team]*	*[what depends on it]*	Planned / In Progress / Complete

Guidance

Common items to decommission after migration or replacement: - **Servers and VMs** — physical or virtual machines no longer needed - **Licences** — software licences that can be released or cancelled - **Network rules** — firewall rules, DNS entries, load balancer configs for retired systems - **Data stores** — databases, file shares, or storage accounts (after data migration and retention period) - **Monitoring and alerting** — dashboards, alerts, and runbooks for retired systems - **Service accounts and credentials** — accounts and secrets for decommissioned integrations - **Documentation** — update or archive architecture documents for retired systems

Technical debt is tracked in Section 6.6 — Technical Debt Register.

Data and Infrastructure Disposal

[Comprehensive]

Attribute	Detail
Data disposal method	*[how data will be securely erased — crypto-shredding, secure wipe, physical destruction]*
Data retention obligations	*[any data that must be retained beyond decommissioning, and for how long]*
Infrastructure disposal	*[how hardware/cloud resources will be decommissioned — resource deletion, subscription cancellation, hardware return]*
Cost savings realised	*[expected cost reduction from decommissioning]*

5.10 Exit Planning

[Recommended]

If hosted in public cloud or with a third-party provider:

Attribute	Detail
Exit strategy	*[how to migrate away from the current provider]*
Data portability	*[how data can be extracted and migrated]*
Vendor lock-in assessment	*[degree of lock-in and mitigation]*
Exit timeline estimate	*[how long an exit would take]*

Scoring Guidance

Score	What This Looks Like
1	CI/CD tool identified but pipeline not documented
3	Development practices, deployment strategy, support model, and release frequency documented; migration plan in place if applicable
5	All of the above plus security scanning integrated in pipeline, team skills assessed with action plan, exit strategy documented with vendor lock-in assessment

6. Decision Making & Governance

[Minimum] `ISO 42010`

Purpose

This section captures the decision-making context for the architecture:

- **Constraints and assumptions** that shaped the design
- **Risks and dependencies** that must be managed
- **Key decisions** made and the rationale behind them
- **Exceptions** to organisational standards
- **Compliance evidence** demonstrating the design has been reviewed and approved

Sub-section	Focus	Depth
6.1 Constraints	Fixed limitations shaping the design	**[Minimum]**
6.2 Assumptions	Factors believed true but not verified	**[Minimum]**
6.3 Risks	Potential events that could affect the design	**[Minimum]**
6.4 Dependencies	External factors the design relies upon	**[Minimum]**
6.5 Issues	Problems that have already materialised	**[Recommended]**
6.6 Technical Debt Register	Debt introduced or inherited	**[Recommended]**
6.7 Guardrail Exceptions	Exceptions to policies and standards	**[Recommended]**
6.8 Architectural Decisions Log	Summary index of ADRs	**[Recommended]**
6.9 Compliance Traceability	Mapping to standards and principles	**[Comprehensive]**
6.10 Approval Sign-Off	Governance approval record	**[Minimum]**

Sections 6.1 through 6.5 collectively form the **RAID log** (Risks, Assumptions, Issues, Dependencies) — a standard project governance tool — extended with **Constraints** to capture fixed limitations. Each element has its own sub-section for clarity, but they should be maintained together as a single, living governance artefact.

6.1 Constraints

[Minimum]

Constraints are fixed limitations that the design must work within. They are non-negotiable and shape the solution boundaries.

ID	Constraint	Category	Impact on Design	Last Assessed
C-001	*[describe the constraint]*	Regulatory / Technical / Commercial / Organisational / Time	*[how this constrains the design]*	*[date]*

Guidance

Common constraints include: - **Regulatory** — Data residency, industry compliance (PCI-DSS, GDPR, FCA) - **Technical** — Must use existing platform, specific technology mandated, legacy integration - **Commercial** — Budget ceiling, existing licensing agreements, vendor contracts - **Organisational** — Team skills, organisational standards, governance processes - **Time** — Fixed delivery deadlines, regulatory compliance dates

6.2 Assumptions

[Minimum]

Assumptions are factors believed to be true but not yet verified. The assumption owner should validate each assumption and track it to closure. An assumption that proves false may become a risk or issue.

ID	Assumption	Impact if False	Certainty	Status	Owner	Evidence
A-001	*[describe the assumption]*	*[impact on design if wrong]*	High / Medium / Low	Open / Closed	*[person]*	*[reference]*

Guidance

Capture assumptions about: - **Technical** — API availability, system capacity, technology compatibility - **Organisational** — Resource availability, team skills, process readiness - **Third-party** — Vendor timelines, SLA commitments, service availability - **Data** — Data quality, volumes, formats, migration feasibility - **Timeline** — Dependent delivery dates, approval timelines

Open assumptions should be tracked actively. The owner is responsible for validating the assumption and providing evidence of closure.

6.3 Risks

[Minimum]

Risks are potential events that could negatively affect the design or its delivery. Each risk should have a mitigation strategy and clear ownership.

Risk identification:

ID	Risk Event	Category	Severity	Likelihood	Owner
R-001	*[describe the risk]*	Technical / Security / Operational / Delivery / Commercial / Compliance	High / Medium / Low	High / Medium / Low	*[person]*

Risk response:

ID	Mitigation Strategy	Mitigation Plan	Residual Risk	Last Assessed
R-001	Accept / Mitigate / Transfer / Avoid	*[details of the mitigation plan]*	High / Medium / Low	*[date]*

Guidance

Consider risks related to: - **Technical** — Technology choices, complexity, integration, performance - **Security** — Vulnerabilities, access control gaps, data exposure - **Operational** — Supportability, monitoring gaps, skill availability - **Delivery** — Timeline, dependencies, resource availability - **Commercial** — Vendor risk, licensing, cost overruns - **Compliance** — Regulatory gaps, standards non-conformance

Mitigation strategies should be one of: **Accept** (business owner sign-off), **Mitigate** (reduce likelihood or impact), **Transfer** (insurance, contractual), or **Avoid** (change the design).

6.4 Dependencies

[Minimum]

Dependencies are external factors that the design relies upon or that rely upon this design. Track both inbound (this design depends on) and outbound (other designs depend on this).

ID	Dependency	Direction	Status	Owner	Evidence	Last Assessed
D-001	*[describe the dependency]*	Inbound / Outbound	Committed / Not Committed / Resolved	*[person]*	*[reference]*	*[date]*

Guidance

Common dependencies include: - **Infrastructure** — Network connectivity, platform availability, shared services - **Application** — APIs, data feeds, upstream/downstream systems - **Team** — Other project deliverables, shared resources, specialist skills - **Vendor** — Third-party service readiness, contract completion - **Governance** — Approval gates, compliance sign-off, change advisory board

6.5 Issues

[Recommended]

Issues are problems that have already materialised and require resolution. Unlike risks (which are potential), issues are current and active.

ID	Issue	Category	Impact	Owner	Resolution Plan	Status	Last Assessed
I-001	*[describe the issue]*	Technical / Security / Operational / Delivery / Commercial	High / Medium / Low	*[person]*	*[resolution plan]*	Open / In Progress / Resolved	*[date]*

Guidance

Issues differ from risks: - A **risk** is something that *might* happen — it is potential - An **issue** is something that *has* happened — it is current and requires active resolution

Common issues include: - **Technical** — A dependency API is not performing as expected; a security vulnerability has been discovered - **Delivery** — A key resource has left the team; a vendor has missed a delivery milestone - **Commercial** — A licence cost has increased beyond budget; a contract negotiation is stalled

Issues should be escalated when they threaten the project timeline, cost, or quality. Track each issue to resolution and record the outcome.

6.6 Technical Debt Register

[Recommended]

Document any technical debt introduced or inherited by this solution. Technical debt can arise at any stage — from initial design shortcuts to inherited legacy constraints — and requires governance tracking alongside risks and issues.

ID	Description	Category	Impact	Remediation Plan	Target Date	Owner
TD-001	*[describe the debt]*	Design / Code / Infrastructure / Security / Operational	High / Medium / Low	*[plan to resolve]*	*[date]*	*[person]*

Guidance

Technical debt should be tracked when: - A shortcut is taken to meet a deadline (document what was deferred and when it will be addressed) - A legacy component is retained that does not

meet current standards - A vendor dependency introduces a constraint that should eventually be removed - A security finding is accepted temporarily with a remediation plan

Each item should have clear ownership and a target date for resolution. Unresolved technical debt often becomes a risk (Section 6.3) or an issue (Section 6.5).

6.7 Guardrail Exceptions

[Recommended]

Document any exceptions to organisational policies, standards, or guardrails:

Policy Exceptions

Question	Response
Does this design create any exception to current policies and standards?	Yes / No
If yes, have exceptions been logged and accepted through the exceptions process?	Yes / No / N/A

Process Exceptions

Question	Response
Does this design conflict with organisational workflows or operating models?	Yes / No
If yes, has this been acknowledged by the process owner?	Yes / No / N/A

Risk Profile Impact

Question	Response
Does this architecture significantly alter the enterprise threat or risk landscape?	Yes / No
If yes, has this been formally assessed by the appropriate security or risk governance authority?	Yes / No / N/A

6.8 Architectural Decisions Log

[Recommended]

Note

This section provides a governance summary index. The full Architecture Decision Records with context, alternatives, and consequences are documented in Section 3.6.2 — Scenarios.

ADR #	Title	Status	Date	Impact
ADR-001	*[title]*	Accepted / Proposed / Superseded	*[date]*	*[brief impact summary]*

6.9 Compliance Traceability

[Comprehensive]

Guidance

This table provides traceability between external standards/principles and the SAD content that demonstrates compliance. It enables reviewers and auditors to verify that all applicable requirements are addressed in the design.

Organisations may use this section to map to their internal design principles, industry standards (e.g., PCI-DSS, ISO 27001), or regulatory requirements.

Map design elements to the standards and principles they satisfy:

Standard / Principle	Requirement	How the Design Satisfies It	Evidence Section
[standard ID]	*[requirement text]*	*[design approach]*	*[section reference]*

Scoring Guidance

Score	What This Looks Like
1	Some risks listed but no mitigation plans or ownership
3	Constraints, assumptions, risks, and dependencies documented with ownership; key ADRs captured; guardrail exceptions logged
5	All of the above plus all items have evidence and dates, compliance traceability complete, issues tracked to resolution

6.10 Approval Sign-Off

[Minimum]

Record who reviewed and approved the architecture:

Role	Name	Date	Decision	Conditions
Solution Architect	*[name]*	*[date]*	Approved / Approved with Conditions / Rejected / Deferred	*[any conditions]*
Security Architect	*[name]*	*[date]*	Approved / Approved with Conditions / Rejected / Deferred	*[any conditions]*

Role	Name	Date	Decision	Conditions
ARB / Design Authority	*[name]*	*[date]*	Approved / Approved with Conditions / Rejected / Deferred	*[any conditions]*
[additional approvers]	*[name]*	*[date]*		

Guidance

The approval sign-off records who reviewed and approved the architecture. Typical approvers include: - **Solution Architect** — the author, confirming the document is complete and accurate - **Security Architect** — confirming the Security View adequately addresses the threat landscape - **ARB / Design Authority** — the governance body confirming the design meets organisational standards - **Business Owner** — confirming the solution meets business requirements (optional but recommended)

Record the approval decision and any conditions that must be met before implementation.

7. Appendices

[Recommended] `ISO 42010`

7.1 Glossary

[Recommended]

Define terms, acronyms, and abbreviations used in this document:

Term	Definition
AD	Architecture Description — ISO 42010 term for the work product expressing an architecture
ADS	Architecture Description Standard — this standard
ADR	Architecture Decision Record
APM	Application Performance Monitoring
ARB	Architecture Review Board — a common form of architecture governance body
AZ	Availability Zone
BYOD	Bring Your Own Device
CDC	Change Data Capture
CDN	Content Delivery Network
CI/CD	Continuous Integration / Continuous Deployment
CISO	Chief Information Security Officer
CMDB	Configuration Management Database
CQRS	Command Query Responsibility Segregation
DAST	Dynamic Application Security Testing
DBA	Database Administrator
DMS	Database Migration Service
DPIA	Data Protection Impact Assessment
DR	Disaster Recovery
EDR	Endpoint Detection and Response
ETL	Extract, Transform, Load
EUC	End User Computing
FaaS	Function as a Service
FinOps	Cloud Financial Operations — a practice for managing cloud costs
HLD	High Level Design — the conceptual-level design content within a SAD (Sections 3–4)
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
IAM	Identity and Access Management

Term	Definition
JDBC	Java Database Connectivity
KMS	Key Management Service
LIA	Legitimate Interests Assessment
mTLS	Mutual Transport Layer Security
NAS	Network Attached Storage
NFR	Non-Functional Requirement
NOC	Network Operations Centre
ODBC	Open Database Connectivity
OIDC	OpenID Connect
PaaS	Platform as a Service
PCI-DSS	Payment Card Industry Data Security Standard
PIA	Privacy Impact Assessment
PII	Personally Identifiable Information
QoS	Quality of Service
RAID	Risks, Assumptions, Issues, Dependencies — a project governance log
RDP	Remote Desktop Protocol
REST	Representational State Transfer — an architectural style for APIs
RPO	Recovery Point Objective — maximum acceptable data loss measured in time
RTO	Recovery Time Objective — maximum acceptable downtime after an incident
SaaS	Software as a Service
SAD	Solution Architecture Document (originally “Software Architecture Document” in RUP)
SAML	Security Assertion Markup Language
SAN	Storage Area Network
SAST	Static Application Security Testing
SCA	Software Composition Analysis
SDLC	Software Development Lifecycle
SFTP	SSH File Transfer Protocol
SIEM	Security Information and Event Management
SLA	Service Level Agreement
SPI	Sensitive Personal Information
SRE	Site Reliability Engineering
SSO	Single Sign-On
TCO	Total Cost of Ownership
TOGAF	The Open Group Architecture Framework
VDI	Virtual Desktop Infrastructure
VPN	Virtual Private Network
WAF (firewall)	Web Application Firewall — a network security control
WAF (framework)	Well-Architected Framework — cloud provider architecture guidance (AWS, Azure, GCP, Oracle, IBM)
[additional terms]	*[definitions]*

Guidance

A glossary ensures shared understanding across all readers. Include: - All acronyms used in the document (even common ones — not everyone knows what RTO means) - Organisation-

specific terminology - Technical terms that may be unfamiliar to non-technical stakeholders - Define terms on first use in the document, and collect them all here for reference

7.2 Reference Documents

[Recommended]

List documents referenced by or related to this SAD:

Document	Version	Description	Location
[document name]	*[version]*	*[what it covers]*	*[link or reference]*

7.3 Standards & Patterns Referenced

[Recommended]

List the standards, design patterns, and principles referenced throughout this document:

Standard / Pattern ID	Name	Version	Applicability
[ID]	*[name]*	*[version]*	*[which sections reference it]*

Templates

Getting a Template

Download a blank SAD template from the **Downloads** page in your preferred format (Word, Markdown, YAML, or JSON).

Three Ways to Use Templates

1. Fill in manually — open the template in your editor, wiki, or word processor and complete each section.

2. Import into an LLM — feed the template to ChatGPT, Claude, or Copilot along with a project brief to generate a structured first draft.

3. Paste into Confluence — copy the Markdown template to your clipboard and paste directly into Confluence, Notion, or any wiki that supports Markdown.

Copy Markdown to Clipboard

Converting Between Formats

Use Pandoc to convert between formats:

```
# Markdown to Word
pandoc sad-template.md -o my-sad.docx

# Markdown to ODF (LibreOffice)
pandoc sad-template.md -o my-sad.odt

# With custom branding
pandoc sad-template.md -o my-sad.docx --reference-doc=your-template.docx
```

Validating Your SAD

ADS is backed by a formal **JSON Schema** — a machine-readable definition of every section, field, enum, and required relationship in a SAD. If you author your SAD in JSON or YAML, you can run an off-the-shelf validator against the schema to confirm the document is structurally complete and the values are within the expected vocabularies (e.g. that `classification` is one of `public / internal / restricted / highly-restricted`).

The validator catches the kinds of mistake that survive a casual review: a missing required field, a typo in an enum value, the wrong type for a field, or a section structurally in the wrong place.

It complements human review — it doesn't replace it.

See **Validating a SAD against the Schema** on the JSON Schema page for the full Node and Python instructions, plus what a validation error looks like and how to read it. Markdown and Word SADs aren't directly validatable in this way, but you can convert them to YAML or JSON via Pandoc or a builder tool and validate the result.

Customising for Your Organisation

You can extend the standard for your organisation without modifying the core sections. There are two extension points:

- `organisationProfile` — maps the standard to your internal tools, standards, and governance gates
- `customSections` — adds organisation-specific sections (e.g., cloud governance checklists, data privacy assessments)

How to Add These to Your SAD

In YAML or JSON: Add the `organisationProfile` and/or `customSections` blocks at the end of your SAD file (after `appendices`). They sit at the top level, alongside the standard sections.

In Markdown: Add new headings after Section 7 (Appendices) with your custom content. Use the same table format as the rest of the document.

In Word: Add new sections after the Appendices. Follow the same heading and table style.

For your whole organisation: Create a customised template with the `organisationProfile` pre-filled and distribute it to your architecture teams. This way every new SAD starts with the correct tool mappings and governance gates.

Organisation Profile Example (YAML)

Add this block to the end of your YAML SAD to map the standard to your organisation's tools and governance:

```yaml
organisationProfile:
  organisationName: "Acme Corp"
  tooling:
    cmdb: "ServiceNow"
    secretStore: "HashiCorp Vault"
    monitoring: "Datadog"
    siem: "Splunk"
    cicd: "GitHub Actions"
    portfolioManagement: "LeanIX"
  internalStandards:
    - id: "STD-001"
      name: "Cloud Platform Standards"
      version: "2.0"
      mappedSections: ["physicalView", "securityView"]
    - id: "POL-008"
      name: "Information Security Policy"
      version: "4.1"
      mappedSections: ["securityView", "dataView"]
```

```yaml
  governanceGates:
    - gateName: "Dev/Test Review"
      requiredMaturityLevel: "minimum"
      requiredSections: ["documentControl", "executiveSummary", "architecturalViews", "riskC
    - gateName: "Production Approval"
      requiredMaturityLevel: "recommended"
      requiredSections: ["documentControl", "executiveSummary", "stakeholders", "architectur
    - gateName: "Enterprise Architecture Review"
      requiredMaturityLevel: "comprehensive"
      requiredSections: ["documentControl", "executiveSummary", "stakeholders", "architectur
```

Organisation Profile Example (JSON)

The same configuration in JSON format:

```json
{
  "organisationProfile": {
    "organisationName": "Acme Corp",
    "tooling": {
      "cmdb": "ServiceNow",
      "secretStore": "HashiCorp Vault",
      "monitoring": "Datadog",
      "siem": "Splunk",
      "cicd": "GitHub Actions",
      "portfolioManagement": "LeanIX"
    },
    "internalStandards": [
      {
        "id": "STD-001",
        "name": "Cloud Platform Standards",
        "version": "2.0",
        "mappedSections": ["physicalView", "securityView"]
      }
    ],
    "governanceGates": [
      {
        "gateName": "Dev/Test Review",
        "requiredMaturityLevel": "minimum",
        "requiredSections": ["documentControl", "executiveSummary", "architecturalViews", "r
      },
      {
        "gateName": "Production Approval",
        "requiredMaturityLevel": "recommended",
        "requiredSections": ["documentControl", "executiveSummary", "stakeholders", "archite
      }
    ]
  }
}
```

Custom Sections Example

Add organisation-specific sections without modifying the core standard:

```yaml
customSections:
  - id: "cloud-governance"
    title: "Cloud Governance Checklist"
    description: "Organisation-specific cloud governance requirements"
    maturityLevel: "recommended"
    parentSection: "physicalView"
    content: |
      | Control | Status | Evidence |
      |---------|--------|----------|
      | Tagging policy applied | Yes / No | |
      | Cost alerts configured | Yes / No | |
      | Backup policy applied | Yes / No | |

  - id: "data-privacy"
    title: "Data Privacy Assessment"
    description: "GDPR and data privacy specific requirements"
    maturityLevel: "recommended"
    parentSection: "dataView"
    content: |
      | Question | Response |
      |----------|----------|
      | DPIA completed? | Yes / No |
      | Data processor agreements in place? | Yes / No |
      | Subject access request process documented? | Yes / No |
```

See Conformance and Usage for full customisation guidelines.

JSON Schema

This page is for developers, tooling builders, and AI practitioners who want to validate, generate, analyse, or process SADs programmatically. If you are writing a SAD manually, you can skip this page and use the templates instead.

Overview

The ADS defines a formal **JSON Schema** (v1.0.0) that enables both authoring and analysis of architecture documents:

Writing: - **AI-assisted authoring** — import the schema into an LLM to generate structured first drafts from a project brief - **Generation** — generate blank templates in multiple formats (JSON, YAML, Markdown, Word) - **Validation** — validate completed SADs against the standard structure

Reading and analysis: - **Programmatic parsing** — read and extract data from SADs using standard JSON/YAML tooling - **Portfolio analysis** — compare SADs across projects to identify coverage gaps, common risks, or technology patterns - **Compliance reporting** — audit SADs at scale against the scoring criteria - **AI-assisted review** — import a completed SAD into an LLM for quality assessment or gap analysis

Extending: - **Extensibility** — add organisation-specific sections via `customSections` and `organisationProfile`

Schema Location

The schema file is located at `/schema/ads.schema.json` in the repository.

Schema ID: `https://archstandard.org/schema/v1.0.0/ads.schema.json`

Schema Structure

The schema maps directly to the standard's sections:

```
ads.schema.json
  schemaVersion          "1.0.0"
  documentControl        Section 0: metadata, history, contributors
  executiveSummary       Section 1: overview, context, scope, decisions
  stakeholders           Section 2: register, concerns, compliance
  architecturalViews     Section 3
     logicalView        3.1: components, services, patterns
     integrationView     3.2: connectivity, integrations, APIs
```

```
    physicalView          3.3: hosting, compute, networking, environments
    dataView              3.4: data stores, classification, privacy
    securityView          3.5: IAM, encryption, monitoring
    scenarios             3.6: use cases, ADRs
  qualityAttributes       Section 4: quality attributes + tradeoffs
  lifecycleManagement     Section 5: SDLC, operations, exit planning
  riskGovernance          Section 6: constraints, risks, decisions, compliance
  appendices              Section 7: glossary, references, approvals
  organisationProfile     Organisation-specific configuration
  customSections[]        Extension point for additional sections
```

Key Schema Features

Documentation Depths

Sections use a `maturityLevel` enum with three values. Note: the schema uses the property name `maturityLevel` while the standard's prose refers to this concept as "Documentation Depth" — they mean the same thing:

```
{ "type": "string", "enum": ["minimum", "recommended", "comprehensive"] }
```

Quality Attribute References

Any section can reference applicable quality attributes:

```
{
  "qualityAttributeRefs": ["reliability", "performance", "cost-optimisation"]
}
```

Organisation Profile

The `organisationProfile` extension allows mapping the generic standard to organisation-specific tools and governance:

```
{
  "organisationProfile": {
    "organisationName": "Acme Corp",
    "tooling": {
      "cmdb": "ServiceNow",
      "secretStore": "HashiCorp Vault",
      "monitoring": "Datadog"
    },
    "governanceGates": [
      {
        "gateName": "Dev/Test Review",
        "requiredMaturityLevel": "minimum",
        "requiredSections": ["documentControl", "executiveSummary", "architecturalViews.logi
      },
      {
        "gateName": "Production Approval",
        "requiredMaturityLevel": "recommended"
      }
```

```
    ]
  }
}
```

Custom Sections

Add organisation-specific sections without modifying the core schema:

```
{
  "customSections": [
    {
      "id": "cloud-governance",
      "title": "Cloud Governance",
      "description": "Organisation-specific cloud governance requirements",
      "content": "...",
      "maturityLevel": "recommended",
      "parentSection": "physicalView"
    }
  ]
}
```

Required vs Optional Fields

The schema enforces only the **Minimum** documentation depth as required:

Required Field	Section
`schemaVersion`	Root
`documentControl.metadata`	0. Document Control
`executiveSummary.solutionOverview`	1. Executive Summary
`architecturalViews`	3. Architectural Views
`riskGovernance.risks`	6. Decision Making & Governance

All other fields are optional, allowing progressive adoption from Minimum through Comprehensive documentation depths.

Validating a SAD against the Schema

This is the canonical home for validation instructions — pages elsewhere on the site (Examples, Templates, FAQ) cross-reference back here so the steps live in one place.

Step 1 — Download the schema

Both `ajv-cli` and Python's `jsonschema` CLI expect a **local file path** for the schema (a remote URL won't work). Download it once and cache it locally:

```
curl -O https://archstandard.org/schema/v1.0.0/ads.schema.json
```

The URL above is the schema's canonical `$id`. The same file is also available via the Downloads page.

Step 2 — Validate your SAD

Using `ajv-cli` (Node)

```
npx -p ajv-cli ajv validate -s ads.schema.json -d my-sad.json
```

The `-p ajv-cli` flag tells `npx` which package to install (the binary is `ajv`, the package is `ajv-cli`).

Using Python `jsonschema`

```
pip install jsonschema
python -m jsonschema -i my-sad.json ads.schema.json
```

Validating a YAML SAD

Convert to JSON first, then pipe through `ajv-cli`:

```
npx js-yaml my-sad.yaml | npx -p ajv-cli ajv validate -s ads.schema.json -d /dev/stdin
```

What validation checks

The schema validates **structure**, **required fields**, `enum` **values**, `pattern` **constraints**, and field types. It does **not** check whether the architecture itself is good — that's what the compliance scoring and reviewer perspectives are for.

Validation passes do not imply governance approval; they confirm that the SAD's structure conforms to the standard.

The schema is hosted at: `https://archstandard.org/schema/v1.0.0/ads.schema.json`

Building Tools with the Schema

The JSON Schema is designed to be consumed by tooling — form builders, validators, converters, and SAD management platforms. Custom extension properties make this easier:

Extension	Purpose	Example
`x-ads-section`	Section number	`"3.1"`, `"6.3"`
`x-ads-title`	Display name	`"Logical View"`, `"Risks"`
`x-ads-depth`	Documentation depth	`"minimum"`, `"recommended"`, `"comprehensive"`

These extensions are present on every major section and sub-section in the schema. A tool can walk the schema tree and use these properties to:

- **Generate forms** with correct section grouping, labels, and depth indicators
- **Show/hide sections** based on the user's chosen documentation depth
- **Validate completeness** against a specific documentation depth
- **Export** to JSON, YAML, Markdown, or DOCX

```
// Example: fetch the schema and list all sections
const schema = await fetch('https://archstandard.org/schema/v1.0.0/ads.schema.json').then(r
```

```
for (const [key, def] of Object.entries(schema.$defs)) {
  if (def['x-ads-section']) {
    console.log(`${def['x-ads-section']} ${def['x-ads-title']} [${def['x-ads-depth']}]`);
  }
}
```

Version History

Versioning Policy

The Architecture Description Standard (ADS) uses **semantic versioning**:

- **MAJOR** version - Structural changes that break backward compatibility (e.g., sections re-organised, removed, or fundamentally changed)
- **MINOR** version - New sections, guidance additions, or non-breaking enhancements
- **PATCH** version - Corrections, clarifications, typo fixes

The **standard version** (e.g. v1.3.0) and the **JSON Schema version** (e.g. v1.0.0) are versioned independently. The schema is the machine-readable contract that SADs and tooling validate against. The schema version only changes when the schema's structure changes; minor and patch releases of the standard add content (guidance, examples, prompts) without touching the schema.

Versions and Schema Compatibility

Every v1.x release of the standard targets JSON Schema v1.0.0 and is published at archstandard.org/v1/. Major versions live at permanent versioned URLs so SADs and tooling targeting a previous version remain valid. A future v2.0.0 of the standard would ship a v2.0.0 schema at `/v2/`, with `/v1/` and the v1.0.0 schema preserved indefinitely.

Version	Status	Notes
v1.3.2	**Current**	Schema consistency fixes and section 6.6/6.8/6.10 additions
v1.3.1	Superseded	Bug fixes and SAD template polish
v1.3.0	Superseded	Adoption release
v1.2.0	Superseded	Public release
v1.1.0	Superseded	Content additions
v1.0.0	Superseded	Initial release; schema established
v2.x	Planned	First schema major bump (v2.0.0)

Compatibility implication: any SAD that validates against schema v1.0.0 is structurally valid against any v1.x.y release of the standard. Editorial improvements (new prompts, new guidance, new examples) do not invalidate prior SADs.

Third-party builders (see specification) should declare which schema version they target.

Releases

Releases are listed newest-first.

v1.3.2 - Schema consistency and Section 6 additions

Date: 2026-04-28 · **JSON Schema:** v1.0.0 (additions only — backwards compatible)

Patch release closing internal-consistency gaps between the schema, the standard text, and the worked examples. All seven examples now validate clean against the schema.

Schema additions (additive — old SADs remain valid): - `riskGovernance.technicalDebt` — array of TD-NNN entries (id, description, category, impact, remediation plan, target date, owner) for **Section 6.6 Technical Debt Register** - `riskGovernance.architectureDecisionsLog` — governance summary index for **Section 6.8 Architectural Decisions Log**, referencing full ADRs in `architecturalViews.scenarios.adrs` (Section 3.6.2) by id - `approvals[].conditions` — optional string capturing conditions on a sign-off

Schema corrections: - New canonical `$defs/classification` (public, internal, restricted, highly-restricted) replaces three inline duplicate enums; removes the stray "confidential" value that had drifted into Document Control - Added `paas-app-service` to physicalView compute types (Cloud Migration example was using it) - External integrations protocol enum brought to parity with internal protocol: added jdbc, odbc, mqtts, websocket, wss, ldaps, ssh; added kerberos to authentication methods (Customer API Platform example was using jdbc)

Structural fix: - `approvals` relocated from `appendices.approvals` to `riskGovernance.approvals`. Per the standard, **Section 6.10 Approval Sign-Off** is the closing governance act of Section 6, not an appendix. Examples migrated; templates and the rendered SAD now correctly place 6.10 under Section 6.

Section-tag corrections: - Compliance Traceability is now correctly tagged 6.9 (was 6.8) - Guardrail-exception fields (policyExceptions, policyExceptionsAccepted, processExceptions, riskProfileImpact) tagged 6.7 - Glossary tagged 7.1, References tagged 7.2

Template / SAD-builder polish: - Section headings in the generated `sad-template.{md,yaml,docx}` now show numbered prefixes for 6.6, 6.7, 6.8, 6.9, 6.10, 7.1, 7.2 — matching the standard

v1.3.1 - Bug fixes and SAD template polish

Date: 2026-04-27 · **JSON Schema:** v1.0.0 (unchanged)

Patch release on top of v1.3.0. No content additions, no breaking changes — bug fixes and editorial polish from initial-review feedback.

Bug fixes: - Fixed broken Mermaid diagram in Medwick example (3.3.1 deployment diagram had a node ID containing spaces after the NHS→MediCore substitution) - Fixed schema canonical URL: now correctly serves at `https://archstandard.org/schema/v1.0.0/ads.schema.json` (matching the `$id`) - Fixed schema validation code samples (must download then validate locally; earlier examples used the unsupported `-s URL` form) - Sidebar scroll-spy: anchor sub-items in single-file standard sections (0, 1, 2, 5, 6, 7) now show active state when the matching section is in view - Consolidated validation instructions to a single canonical home on the JSON Schema page (was duplicated across 4 pages)

SAD template improvements: - Wide multi-column tables (e.g. Data Stores) now render as vertical Field/Value forms — readable in any output format - Word doc margins reduced to 15mm; `lang: en-GB` set so Word's spell-checker uses UK English - "Author: Andi Chandler" removed from auto-generated headers (template is for the user to fill in) - Acronym dictionary expanded: SRE, DevOps, MLOps, DBA, CPU, RHEL, SUSE, CentOS, QA, CyberArk, CI/CD, Trade-off - "Quality Attribute Refs" → "Quality Attribute References" (full word) - Banner items now render as separate paragraphs in the Word doc

Editorial polish: - Stale NHS-flavoured Medwick descriptions updated across the site (now uses fictional "MediCore" framework consistently, including in the Medwick logo SVG)

v1.3.0 - Adoption Release

Date: 2026-04-24 · **JSON Schema:** v1.0.0 (unchanged)

Focus on making ADS easier to adopt, easier to review, and easier to teach.

New example SADs (four → seven): - **NorthWind Retail** — Recommended depth, Tier 2, PCI-DSS regulated e-commerce on AWS - **Medwick Healthcare** — Comprehensive depth, Tier 1, fictional national-healthcare patient portal with clinical safety standards, healthcare DSPT-equivalent governance, and FHIR R4 integration - **Stellar Platform** — Recommended depth, Tier 3, multi-cloud Internal Developer Platform on Kubernetes with Backstage

AI Prompt Library (new): - First-draft generator — produce a structured SAD from a brief - Validator — completeness, consistency, clarity, credibility - Scorer — apply the 0-5 compliance scale with justification - Improver — section-by-section suggestions for the next score level - Security review — CISO-office-style review - Governance review — ARB-style review

Guidance section (new): - What Good Looks Like — worked excerpts showing high-quality content - Anti-Patterns — common mistakes with before-and-after examples - Decision Guides — flowcharts for depth, threat model, split SADs, ADRs, RAID classification - Reviewer Perspectives — what ARB chair, Security, Data, SRE, Finance, Change, Product look for - Starter Kits — pre-scoped guidance for new cloud apps, migrations, integrations, platforms - Review Checklist — printable one-pager for governance reviewers - Industry Mappings — GDS Service Standard, NIST CSF, PCI-DSS, ISO 27001, NHS DSPT, UK GDPR, FCA - Cheat Cards — one-page printable references - The 2-Minute Pitch — speaker notes for introducing ADS internally

Contribution infrastructure: - CONTRIBUTING.md with style guide and contribution paths - CODE_OF_CONDUCT.md (principles-based) - GitHub issue templates for bug reports, feature requests, and example contributions

Other: - FAQ page answering common adoption questions - Examples index expanded to show all seven examples with downloadable Markdown and JSON

v1.2.0 - Public Release

Date: 2026-04-13 · **JSON Schema:** v1.0.0 (unchanged)

- Public launch of archstandard.org and GitHub repository
- All example SADs available in JSON and Markdown download formats
- Reading preferences toolbar (font choice, line spacing, focus mode, colour overlay)
- Security hardening (CSP headers, Mermaid strict mode, dependency updates)
- Depth Cheat Sheet for quick reference
- Quickstart guide
- Organisation customisation examples (YAML and JSON)
- Full French and German translation parity (30 pages each)

- Dual licence: CC BY 4.0 (content) + MIT (code)

v1.1 - Examples and Guidance

Date: 2026-03-31 · **JSON Schema:** v1.0.0 (unchanged)

- Four completed example SADs (Employee Directory, Customer API Platform, Cloud Migration, archstandard.org)
- Scoring guidance (1/3/5) added to all section pages
- Accessibility and neurodiversity improvements
- French and German translations
- Template generator script (schema as single source of truth)

v1.0.0 - Initial Release

Date: 2026-03-27 · **JSON Schema:** v1.0.0

The first version of the Architecture Description Standard (ADS), synthesising years of practical experience in solution architecture with established frameworks:

- **ISO/IEC/IEEE 42010** - Architecture Description meta-model (stakeholders, concerns, viewpoints, views)
- **4+1 Architectural View Model** - Six architectural views (extended from 5 with dedicated Data and Security views)
- **Cloud Well-Architected Frameworks** - Quality attributes derived from AWS, Azure, GCP, Oracle, and IBM Well-Architected Frameworks
- **TOGAF** - Architecture domain alignment (Business, Data, Application, Technology)
- **Industry SAD/HLD templates** - Best practices from arc42, ServiceNow, NHS SAF, ALMBoK, and enterprise templates

Structure: - 8 top-level sections (0-7) - 6 Architectural Views - 5 Quality Attributes - 3 Documentation Depths (Minimum, Recommended, Comprehensive) - JSON Schema for multi-format generation - Framework alignment traceability

Roadmap

Future versions may include:

- **v2.0** — Community feedback incorporation and structural refinements. A v2.0 release would, for the first time, ship a new schema (v2.0.0) alongside it. The current schema (v1.0.0) and any SADs validated against it would remain available at `/v1/` permanently.