IEEE 29148:2018 Software Requirements Specification

Model Context Protocol (MCP) Server

Mark Sigler

February 25, 2026

Contents

Chapter 1

Software Requirements Specification: MCP Server

Document Identifier: MCP-SRS-001
Version: 1.1.0
Date: 2025-07-17
Standard: ISO/IEC/IEEE 29148:2018
Status: Approved
Author: Mark Sigler

1.1 Document Control

Version	Date	Author	Changes
1.1.0	2025-07-17	Mark Sigler	Aligned with MCP specification 2025-11-25: elevated Streamable HTTP to MUST, added session management (FR-PROTO-025–031), lifecycle management (FR-PROTO-032–034), resource annotations/list_changed (FR-RSRC-014–016), tool outputSchema/annotations/list_changed (FR-TOOL-023–025), prompt enhancements (FR-PROMPT-007–010), expanded Task requirements (FR-TASK-004–012), MCP security best practices (NFR-SEC-073–081)
1.0.0	2026-02-23	Mark Sigler	Initial SRS derived from MCP-PRD v2.4, structured per IEEE 29148:2018

1.2 Table of Contents

1.3 1. Introduction

1.3.1 1.1 Purpose

This Software Requirements Specification (SRS) defines all functional and non-functional requirements for a production-ready Model Context Protocol (MCP) server. The MCP server enables

Large Language Models (LLMs) to securely access and interact with external data sources and tools via the open MCP specification.

This document is structured per ISO/IEC/IEEE 29148:2018 and uses EARS (Easy Approach to Requirements Syntax) patterns for unambiguous requirement statements (see Requirements Engineering Standards).

1.3.2 1.2 Scope

The MCP server is a lightweight, containerized service exposing capabilities through three core primitives:

Primitive	Control Model	Description
Resources	Application-controlled	Passive data sources providing read-only context
Tools	Model-controlled	Functions the LLM can call to perform actions
Prompts	User-controlled	Pre-built instruction templates for specific workflows

In scope:

- MCP server implementation compliant with MCP specification 2025-11-25
- Resources, tools, and prompts for a target integration domain
- OAuth 2.1 authorization and JWT authentication for HTTP transport
- Containerized deployment via Docker and container registries
- Portability across all MCP-compliant clients
- Distribution via MCP Registry and sub-registries
- AI service provider agnostic deployment
- Enterprise security, monitoring, and observability

Out of scope:

- Custom MCP client development
- Modifications to the MCP protocol specification
- Legacy system modernization (integration only)
- End-user application development
- AI provider-specific custom features

1.3.3 1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
MCP	Model Context Protocol — open standard for AI-system integration
JSON-RPC	JSON Remote Procedure Call protocol used for MCP communication
Resource	A data source accessible via URI (application-controlled primitive)

Term	Definition
Tool	An executable function exposed by the server (model-controlled primitive)
Prompt	A reusable template with optional parameters (user-controlled primitive)
Sampling	Server-initiated request for LLM completion from client
Elicitation	Server-initiated request for user input through the client
Task	Durable request with polling and deferred result retrieval (experimental)
Primitive	Core MCP capability type (resource, tool, prompt)
PKCE	Proof Key for Code Exchange — required for OAuth 2.1 public clients
JWT	JSON Web Token — bearer authentication token
JWKS	JSON Web Key Set — public keys for JWT verification
EARS	Easy Approach to Requirements Syntax
RBAC	Role-Based Access Control
SSE	Server-Sent Events
OIDC	OpenID Connect
SBOM	Software Bill of Materials
SRS	Software Requirements Specification

1.3.4 1.4 References

Normative References:

Reference	URL
MCP Specification (2025-11-25)	https://modelcontextprotocol.io/docs/
MCP Specification Changelog	https://modelcontextprotocol.io/specification/2025-11-25/changelog
MCP Authorization Tutorial	https://modelcontextprotocol.io/docs/tutorials/security/authorization
MCP Server Concepts	https://modelcontextprotocol.io/docs/learn/server-concepts
MCP Build Server Guide	https://modelcontextprotocol.io/docs/develop/build-server
MCP Registry	https://registry.modelcontextprotocol.io

Reference	URL
JSON-RPC 2.0 Specification	https://www.jsonrpc.org/specification
JSON Schema 2020-12	https://json-schema.org/
OAuth 2.1	https://oauth.net/2.1/
RFC 6570 (URI Templates)	https://tools.ietf.org/html/rfc6570
RFC 7636 (PKCE)	https://tools.ietf.org/html/rfc7636
RFC 9728 (Protected Resource Metadata)	https://datatracker.ietf.org/doc/html/rfc9728
CIS Docker Benchmark	https://www.cisecurity.org/benchmark/docker
12-Factor App	https://12factor.net/

Informative References:

Reference	Path
IEEE 42010 Architecture Description	../IEEE-42010/AD.md
Requirements Engineering Standards	../02b-requirements-engineering.md
Architecture Decision Records	../01b-architecture-decisions.md

1.3.5 1.5 Document Overview

- **§2** identifies stakeholders, their concerns, and the core principles constraining the system
- **§3** specifies all functional requirements (FR-xxx) and non-functional requirements (NFR-xxx) using EARS syntax
- **§4** defines design constraints and technology mandates
- **§5** maps each requirement to a verification method
- **§6** provides a bidirectional traceability matrix linking requirements to architecture sections and test categories

---

1.4 2. Stakeholders and Concerns

1.4.1 2.1 Stakeholder Identification

Stakeholder	Role	Primary Concerns
Product Owner	Decision authority	Feature prioritization, roadmap alignment, business value
Engineering Lead	Technical authority	Architecture compliance, technical feasibility

Stakeholder	Role	Primary Concerns
Security Lead	Security authority	Risk assessment, compliance verification, threat mitigation
DevOps Lead	Operations authority	Deployment, monitoring, reliability, scaling
Enterprise Architect	Standards authority	Integration patterns, technology governance

1.4.2 2.2 Primary Users

User Type	Description	Key Needs
Enterprise Developers	Building AI-powered applications	Secure data access, clear documentation, debugging tools
Data Scientists	Creating AI workflows with multiple data sources	Easy integration, reliable performance, data access
Platform Engineers	Managing MCP server deployments	Scalability, monitoring, deployment automation
Security Engineers	Ensuring compliance and security	Audit trails, access controls, vulnerability management

1.4.3 2.3 Core Principles (Foundational Constraints)

The following principles are **mandatory constraints** governing all requirements in this SRS.

1.4.3.1 CP-01: Client Portability

The MCP server shall function with any MCP-compliant client without modification. No client-specific code or dependencies shall exist. The server shall be transport-agnostic (stdio for local development, HTTP/SSE for production).

Applicable clients: Claude Desktop, ChatGPT Desktop, GitHub Copilot, Cursor, VS Code, JetBrains IDEs, custom enterprise clients, any open-source MCP client.

1.4.3.2 CP-02: Registry Distribution

The MCP server shall be distributable via the MCP Registry (`registry.modelcontextprotocol.io`) and compatible sub-registries. Complete `server.json` metadata, container images on standard registries (ghcr.io, Docker Hub), and MCP moderation guideline compliance are required.

1.4.3.3 CP-03: AI Service Provider Agnostic

The MCP server shall be deployable with any AI service provider (AWS Bedrock, Azure OpenAI, Google Vertex AI, OpenAI, Anthropic, vLLM, Ollama, custom). No hardcoded provider dependencies. Provider selection via configuration. Provider authentication via environment variables.

1.4.3.4 CP-04: Separation of Concerns

Each MCP server shall focus on a single integration domain with cohesive capabilities. All tools, resources, and prompts within a server shall share a common domain context. Cross-domain workflows shall be achieved by clients orchestrating multiple servers.

1.4.4 2.4 Success Metrics

1.4.4.1 Technical Metrics

Metric	Target
Response Time (p95)	< 500ms
Availability	99.9% uptime
Error Rate	< 0.1%
Container Startup Time	< 5 seconds
Security Vulnerabilities	Zero critical/high
Test Coverage	> 80% overall

1.4.4.2 Business Metrics

Metric	Target
Client Integrations	3+ different MCP clients verified
Developer Setup Time	< 30 minutes
Active Deployments (90 days)	10+
User Satisfaction	80%+ positive
Security Incidents	Zero unauthorized access

1.5 3. Specific Requirements

Requirements use EARS syntax patterns:

- **Ubiquitous:** `The <system> shall <response>`
- **Event-driven:** `When <trigger>, the <system> shall <response>`
- **State-driven:** `While <state>, the <system> shall <response>`
- **Conditional:** `If <condition>, the <system> shall <response>`
- **Optional:** `Where <feature is supported>, the <system> shall <response>`

Priority levels: **MUST** (mandatory), **SHOULD** (expected), **MAY** (optional), **EXPERIMENTAL** (subject to change).

1.5.1 3.1 External Interface Requirements

1.5.1.1 3.1.1 Transport Interfaces

ID	Requirement	Priority
FR-PROTO-001	The MCP server shall support Streamable HTTP transport as the primary HTTP transport for production deployments, accepting JSON-RPC messages via HTTP POST and optionally upgrading responses to Server-Sent Events (SSE) streams.	MUST
FR-PROTO-002	The MCP server should support backward compatibility with the deprecated HTTP+SSE transport (separate SSE endpoint) for clients that have not migrated to Streamable HTTP.	SHOULD
FR-PROTO-003	Where local development is the deployment context, the MCP server shall support stdio transport.	SHOULD
FR-PROTO-004	The MCP server shall implement JSON-RPC 2.0 as the message protocol over all transports.	MUST
FR-PROTO-005	While operating with HTTP transport, the MCP server shall require TLS 1.2 or higher encryption.	MUST
FR-PROTO-006	When receiving an HTTP request with an invalid Origin header, the MCP server shall respond with HTTP 403 Forbidden to prevent DNS rebinding attacks.	MUST
FR-PROTO-006a	When operating as a local server, the MCP server shall bind exclusively to the loopback interface (localhost/127.0.0.1).	MUST
FR-PROTO-007	While operating with stdio transport, the MCP server shall write only JSON-RPC messages to stdout and all other output to stderr.	MUST
FR-PROTO-008	When a connection is established, the MCP server shall complete the connection handshake within 2 seconds.	MUST
FR-PROTO-009	When receiving a malformed JSON-RPC message, the MCP server shall return a structured error response without crashing.	MUST
FR-PROTO-010	When shutting down, the MCP server shall perform clean resource cleanup with no orphaned processes.	MUST

1.5.1.2 3.1.1a Streamable HTTP Session Management

ID	Requirement	Priority
FR-PROTO-025	When the MCP server assigns a session, the server shall return an `MCP-Session-Id` header in the initialization response, and the client shall include this header in all subsequent requests.	MUST
FR-PROTO-026	When the MCP server receives a request with an invalid or expired `MCP-Session-Id`, the server shall respond with HTTP 404 Not Found so the client can re-initialize.	MUST
FR-PROTO-027	The MCP server shall include the `MCP-Protocol-Version` header (set to the negotiated protocol version) in all HTTP responses after initialization.	MUST
FR-PROTO-028	When a client reconnects to an SSE stream, the MCP server should support resumability by accepting the `Last-Event-ID` header and redelivering missed events.	SHOULD
FR-PROTO-029	The MCP server shall support HTTP GET requests on the MCP endpoint for clients to open an SSE stream for server-initiated messages (notifications and requests).	SHOULD
FR-PROTO-030	The MCP server shall implement timeout handling for all requests, with per-request configurability and optional progress-based timeout reset.	MUST
FR-PROTO-031	The MCP server shall support `ping` requests from clients for connection keepalive, responding immediately.	MUST

1.5.1.3 3.1.1b Lifecycle Management

ID	Requirement	Priority
FR-PROTO-032	The MCP server shall not send any requests to clients until receiving the `initialized` notification from the client.	MUST
FR-PROTO-033	When the MCP server receives a `notifications/cancelled` message, the server shall stop the referenced request and free associated resources, returning no response for that request.	MUST
FR-PROTO-034	The MCP server shall not cancel an in-flight `initialize` request.	MUST

1.5.1.4 3.1.2 Client Interface

ID	Requirement	Priority
FR-PROTO-011	The MCP server shall function identically across all MCP-compliant clients without client-specific code.	MUST
FR-PROTO-012	The MCP server shall declare supported capabilities (resources, tools, prompts, sampling, elicitation) during the initialization handshake.	MUST
FR-PROTO-013	The MCP server shall provide server metadata including name, version, and optional description during initialization.	MUST
FR-PROTO-014	The MCP server shall support protocol version negotiation per MCP specification 2025-11-25.	MUST
FR-PROTO-015	The MCP server shall parse and validate 100% of messages as valid JSON-RPC 2.0.	MUST

1.5.1.5 3.1.3 Registry Interface

ID	Requirement	Priority
FR-PROTO-016	The MCP server shall provide complete `server.json` metadata for MCP Registry listing.	MUST
FR-PROTO-017	The MCP server shall comply with MCP moderation guidelines for registry distribution.	MUST
FR-PROTO-018	The MCP server shall support registry API schema compatibility for discovery by clients.	MUST

1.5.2 3.2 Functional Requirements

1.5.2.1 3.2.1 Resource Management

ID	Requirement	Priority
FR-RSRC-001	The MCP server shall implement the `resources/list` endpoint returning all accessible resources with URI, name, description, and MIME type.	MUST
FR-RSRC-002	The MCP server shall implement the `resources/read` endpoint returning resource content by URI.	MUST

ID	Requirement	Priority
FR-RSRC-003	When the resource set exceeds 100 items, the MCP server shall support cursor-based pagination.	MUST
FR-RSRC-004	The MCP server shall support both text and binary (base64-encoded) resource content types.	MUST
FR-RSRC-005	When an invalid or nonexistent URI is provided to `resources/read`, the MCP server shall return error code `-32002` (Resource Not Found) with a descriptive message.	MUST
FR-RSRC-006	The MCP server shall handle resources larger than 10MB efficiently via streaming.	MUST
FR-RSRC-007	The MCP server shall implement `resources/templates/list` returning URI template definitions with metadata.	SHOULD
FR-RSRC-008	The MCP server shall support URI template expansion with variable substitution per RFC 6570.	SHOULD
FR-RSRC-009	Where resource templates are supported, the MCP server shall provide parameter completion suggestions as the user types.	SHOULD
FR-RSRC-010	Where resource subscriptions are supported, the MCP server shall implement `resources/subscribe` and `resources/unsubscribe` endpoints.	MAY
FR-RSRC-011	When a subscribed resource changes, the MCP server shall send `notifications/resources/updated` within 5 seconds.	MAY
FR-RSRC-012	When a client disconnects, the MCP server shall clean up all associated subscriptions with no memory leaks.	MAY
FR-RSRC-013	The MCP server shall return resource listing responses within 200ms for typical resource counts.	MUST
FR-RSRC-014	The MCP server shall support optional `title` and `size` fields on resource definitions for enhanced client display.	SHOULD
FR-RSRC-015	The MCP server shall support resource `annotations` including `audience` (list of `user` or `assistant`), `priority` (0.0–1.0), and `lastModified` (ISO 8601 timestamp).	SHOULD

ID	Requirement	Priority
FR-RSRC-016	Where the `listChanged` capability is declared, the MCP server shall send `notifications/resources/list_changed` when the set of available resources changes.	MUST

1.5.2.2 3.2.2 Tool Execution

ID	Requirement	Priority
FR-TOOL-001	The MCP server shall implement the `tools/list` endpoint returning tool definitions with name, description, and input schema.	MUST
FR-TOOL-002	The MCP server shall define tool input schemas using JSON Schema 2020-12.	MUST
FR-TOOL-003	The MCP server shall implement the `tools/call` endpoint executing the requested tool with validated arguments.	MUST
FR-TOOL-004	When tool arguments fail schema validation, the MCP server shall return a Tool Execution Error (result with `isError: true`) rather than a Protocol Error.	MUST
FR-TOOL-005	The MCP server shall validate all tool inputs against declared schemas before execution.	MUST
FR-TOOL-006	The MCP server shall support text, binary (image), audio, and resource link content types in tool results.	MUST
FR-TOOL-007	The MCP server shall support at least 5 simultaneous tool executions concurrently.	MUST
FR-TOOL-008	Each tool name shall be 1–128 characters, case-sensitive, using only `[A-Za-z0-9_\-./]`, following verb_noun convention (e.g., `get_forecast`, `create_issue`).	MUST
FR-TOOL-009	Each tool shall include clear description, argument documentation with types, and at least one usage example.	MUST
FR-TOOL-010	Where icon support is implemented, the MCP server shall expose an optional `icons` array (with URI and media type) for tools, resources, resource templates, and prompts.	SHOULD
FR-TOOL-011	When a tool integrates with an external API, the MCP server shall include a descriptive User-Agent header (e.g., `mcp-server/1.0`).	MUST

ID	Requirement	Priority
FR-TOOL-012	When a tool integrates with an external API, the MCP server shall enforce an explicit timeout (default 30 seconds, configurable).	MUST
FR-TOOL-013	When a tool encounters a failure, the MCP server shall return a user-friendly error message without exposing internal details, stack traces, or sensitive data.	MUST
FR-TOOL-014	When a tool integrates with an external API, the MCP server shall implement exponential backoff for transient failures.	MUST
FR-TOOL-015	When a tool integrates with an external API, the MCP server shall implement the circuit breaker pattern to prevent cascade failures.	MUST
FR-TOOL-016	When a tool operation exceeds 5 seconds, the MCP server shall send progress notifications at least every 5 seconds.	SHOULD
FR-TOOL-017	The MCP server shall support cancellation of in-flight tool executions via `notifications/cancelled`, stopping execution within 2 seconds and freeing associated resources.	SHOULD
FR-TOOL-018	The MCP server shall support configurable per-tool timeout handling.	SHOULD
FR-TOOL-023	Where the `listChanged` capability is declared, the MCP server shall send `notifications/tools/list_changed` when the set of available tools changes.	MUST
FR-TOOL-024	Where structured output is required, the MCP server shall support `outputSchema` on tool definitions using JSON Schema 2020-12 and return matching `structuredContent` in tool results alongside backward-compatible `content`.	SHOULD
FR-TOOL-025	The MCP server shall support tool annotations (`readOnlyHint`, `destructiveHint`, `idempotentHint`, `openWorldHint`) to convey tool behavior metadata to clients.	SHOULD

1.5.2.3 3.2.3 Tool Consent and Safety

ID	Requirement	Priority
FR-TOOL-019	The MCP server shall expose all available tools for display in client UIs.	MUST

ID	Requirement	Priority
FR-TOOL-020	The MCP server shall log all tool executions with user identity, timestamp, parameters, and result for audit trails.	MUST
FR-TOOL-021	The MCP server shall support consent level metadata for tools: always-ask, ask-once, pre-approved, and disabled.	SHOULD
FR-TOOL-022	The MCP server shall support per-user tool enable/disable configuration.	SHOULD

1.5.2.4 3.2.4 Prompt Management

ID	Requirement	Priority
FR-PROMPT-001	The MCP server shall implement the `prompts/list` endpoint returning prompt templates with metadata and argument schemas.	MUST
FR-PROMPT-002	The MCP server shall implement the `prompts/get` endpoint returning formatted prompt content with argument interpolation.	MUST
FR-PROMPT-003	When a prompt references resources, the MCP server shall resolve and embed the referenced resource content.	SHOULD
FR-PROMPT-004	When optional arguments are missing from a `prompts/get` request, the MCP server shall apply graceful defaults.	MUST
FR-PROMPT-005	The MCP server shall provide parameter completion suggestions for prompt argument values.	SHOULD
FR-PROMPT-006	Each prompt shall include a clear description and usage examples.	MUST
FR-PROMPT-007	The MCP server shall support optional `title` and `icons` fields on prompt definitions for enhanced client display.	SHOULD
FR-PROMPT-008	The MCP server shall support audio content (with `data` and `mimeType`) in prompt messages alongside text and image content.	SHOULD
FR-PROMPT-009	When an invalid prompt name or missing required arguments are provided, the MCP server shall return error code `-32602` (Invalid Params) with a descriptive message.	MUST

ID	Requirement	Priority
FR-PROMPT-010	Where the `listChanged` capability is declared, the MCP server shall send `notifications/prompts/list_changed` when the set of available prompts changes.	MUST

1.5.2.5 3.2.5 Sampling

ID	Requirement	Priority
FR-SAMP-001	Where sampling is supported, the MCP server shall implement the request/response pattern for LLM completions per the MCP sampling specification.	MAY
FR-SAMP-002	Where sampling is supported, the MCP server shall support completion parameters (temperature, max tokens).	MAY
FR-SAMP-003	When a client denies a sampling request, the MCP server shall degrade gracefully without failure.	MAY
FR-SAMP-004	Where sampling is supported, the MCP server shall support `tools` and `toolChoice` parameters enabling tool calling within sampling requests.	MAY
FR-SAMP-005	Where sampling is supported, the MCP server shall respect declared token limits.	MAY

1.5.2.6 3.2.6 Elicitation

ID	Requirement	Priority
FR-ELIC-001	Where elicitation is supported, the MCP server shall request user input during tool execution or workflow via the `elicitation/create` method.	MAY
FR-ELIC-002	Where elicitation is supported, the MCP server shall support input types: string, number, enum (single and multi-select), and URL.	MAY
FR-ELIC-003	Where elicitation is supported, the MCP server shall support default values for all primitive input types.	MAY
FR-ELIC-004	Where elicitation is supported, the MCP server shall support titled and untitled enum variants.	MAY

ID	Requirement	Priority
FR-ELIC-005	Where elicitation is supported, the MCP server shall present clear UI for user input requests through the client.	MAY

1.5.2.7 3.2.7 Tasks (Experimental)

ID	Requirement	Priority
FR-TASK-001	Where tasks are supported, the MCP server shall track durable requests with unique task identifiers assigned at request receipt.	EXPERIMENTAL
FR-TASK-002	Where tasks are supported, the MCP server shall implement `tasks/get` for polling-based task status retrieval with configurable intervals.	EXPERIMENTAL
FR-TASK-003	Where tasks are supported, the MCP server shall implement `tasks/result` for deferred retrieval of completed task results.	EXPERIMENTAL
FR-TASK-004	Where tasks are supported, the MCP server shall manage task status through the defined lifecycle: `working` → `completed` / `failed` / `cancelled` / `input_required`.	EXPERIMENTAL
FR-TASK-005	Where tasks are supported, the MCP server shall include the `task` field (with `id` and `status`) in responses and notifications for task-augmented requests.	EXPERIMENTAL
FR-TASK-006	Where tasks are supported, the MCP server shall send `notifications/tasks/progress` with current task status and optional `progress` percentage for long-running operations.	EXPERIMENTAL
FR-TASK-007	Where tasks are supported, the MCP server shall implement `tasks/cancel` allowing clients to request cancellation of in-progress tasks, transitioning status to `cancelled`.	EXPERIMENTAL
FR-TASK-008	Where tasks are supported, the MCP server shall implement `tasks/list` returning active tasks for the current session.	EXPERIMENTAL
FR-TASK-009	Where tasks are supported, the MCP server shall enforce a configurable time-to-live (TTL) on task results and free associated resources after expiry.	EXPERIMENTAL

ID	Requirement	Priority
FR-TASK-010	Where tasks are supported, tool definitions shall declare task behavior via `execution.taskSupport` with values `required`, `optional`, or `forbidden` (default `forbidden`).	EXPERIMENTAL
FR-TASK-011	Where tasks are supported, the MCP server shall isolate task data between clients and enforce access control so that only the originating client can retrieve or cancel a task.	EXPERIMENTAL
FR-TASK-012	Where tasks are supported, the MCP server shall log all task lifecycle transitions (creation, status changes, completion, cancellation) for audit purposes.	EXPERIMENTAL

Note: Task requirements are experimental per MCP specification 2025-11-25 and may change in future versions.

1.5.2.8 3.2.8 Multi-Server Orchestration

ID	Requirement	Priority
FR-ORCH-001	The MCP server shall declare clear capability boundaries for orchestration with other MCP servers.	SHOULD
FR-ORCH-002	When multiple MCP servers are composed by a client, resources from different servers shall be combinable.	SHOULD
FR-ORCH-003	When multiple MCP servers are composed by a client, tools from different servers shall be usable together.	SHOULD
FR-ORCH-004	When one server in a multi-server composition fails, other servers shall continue operating without impact.	SHOULD
FR-ORCH-005	The MCP server shall not access other servers' data directly; cross-server workflows shall be orchestrated by clients.	MUST

1.5.2.9 3.2.9 AI Service Provider Gateway

ID	Requirement	Priority
FR-GWWY-001	The gateway shall provide a configurable `base_url` parameter to redirect API calls to any OpenAI-compatible endpoint.	SHOULD

ID	Requirement	Priority
FR-GWWY-002	When an administrator saves a new endpoint, the gateway shall execute an automated `/v1/models` handshake to verify connectivity.	SHOULD
FR-GWWY-003	While an internal proxy is configured, the gateway shall inject mandatory enterprise headers (e.g., `X-Project-ID`, `X-Cost-Center`) into every request.	SHOULD
FR-GWWY-004	If the primary provider returns a 429 (Rate Limit) error, the gateway shall trigger automatic failover to the secondary provider.	SHOULD
FR-GWWY-005	If the primary provider returns a 5xx error, the gateway shall implement exponential backoff (1s, 2s, 4s, 8s) then failover to secondary.	SHOULD
FR-GWWY-006	If the primary provider times out, the gateway shall immediately failover to the secondary provider.	SHOULD
FR-GWWY-007	The gateway shall store all LLM provider credentials in an encrypted secret management service, never in plain-text config files.	MUST
FR-GWWY-008	The gateway shall support model ID mapping from application model IDs to upstream provider model IDs.	SHOULD
FR-GWWY-009	The gateway shall enforce per-instance rate limits (tokens per minute, requests per minute).	SHOULD
FR-GWWY-010	The gateway shall track usage by cost center, project, and environment.	SHOULD
FR-GWWY-011	The gateway shall support automatic credential rotation without service restart.	SHOULD
FR-GWWY-012	Where the MCP server implements sampling, the server shall route LLM requests through the gateway using the configured endpoint and authentication.	SHOULD

1.5.2.10 3.2.10 Error Handling

ID	Requirement	Priority
FR-PROTO-019	The MCP server shall use standard JSON-RPC 2.0 error codes: -32700 (Parse Error), -32600 (Invalid Request), -32601 (Method Not Found), -32602 (Invalid Params), -32603 (Internal Error).	MUST
FR-PROTO-020	When a tool input validation error occurs, the MCP server shall return it as a Tool Execution Error (`result.isError: true`) not a Protocol Error.	MUST
FR-PROTO-021	The MCP server shall not expose internal implementation details, stack traces, or sensitive data in any error response.	MUST
FR-PROTO-022	When a transient error occurs, the MCP server should include retry guidance (retry-after, backoff hints) in the error response.	SHOULD
FR-PROTO-023	The MCP server shall include a correlation ID in every error response for traceability.	MUST
FR-PROTO-024	The MCP server shall log all error responses with full context (correlation ID, user, endpoint, parameters) for debugging.	MUST

1.5.3 3.3 Non-Functional Requirements

1.5.3.1 3.3.1 Security — Authorization Framework

ID	Requirement	Priority
NFR-SEC-001	While operating with HTTP transport, the MCP server shall implement OAuth 2.1 authorization with mandatory PKCE.	MUST
NFR-SEC-002	The MCP server shall not support the OAuth implicit flow.	MUST
NFR-SEC-003	The MCP server shall support OpenID Connect Discovery 1.0 for authorization server metadata.	MUST
NFR-SEC-004	The MCP server shall expose a `/.well-known/oauth-protected-resource` endpoint returning Protected Resource Metadata per RFC 9728.	MUST
NFR-SEC-005	The MCP server shall support OAuth Client ID Metadata Documents or Dynamic Client Registration (DCR) for client registration.	MUST

ID	Requirement	Priority
NFR-SEC-006	The MCP server shall support incremental scope consent via `WWW-Authenticate` header.	MUST
NFR-SEC-007	When receiving a request without a valid token, the MCP server shall respond with `401 Unauthorized` and `WWW-Authenticate` header pointing to Protected Resource Metadata.	MUST
NFR-SEC-008	The MCP server shall validate bearer tokens: signature, issuer (`iss`), audience (`aud`), and expiration (`exp`).	MUST
NFR-SEC-009	The MCP server shall support per-capability scope validation (`mcp:tools`, `mcp:resources`, `mcp:prompts`).	MUST

1.5.3.2 3.3.2 Security — Authentication

ID	Requirement	Priority
NFR-SEC-010	The MCP server shall support JWT/JWKS authentication with signature verification using cached public keys.	MUST
NFR-SEC-011	The MCP server shall support OAuth 2.1 authentication for HTTP transport.	MUST
NFR-SEC-012	The MCP server shall support API key authentication for service-to-service access.	SHOULD
NFR-SEC-013	The MCP server shall cache JWKS keys with a configurable TTL (default 3600 seconds).	MUST
NFR-SEC-014	The MCP server shall support configurable clock skew tolerance (default 60 seconds) for token validation.	MUST
NFR-SEC-015	The MCP server shall use an access token TTL of 900 seconds (15 minutes) and refresh token TTL of 86400 seconds (24 hours) as defaults.	MUST
NFR-SEC-016	When an invalid token is presented, the MCP server shall respond with 401 and a clear error message.	MUST

1.5.3.3 3.3.3 Security — Authorization and Access Control

ID	Requirement	Priority
NFR-SEC-017	The MCP server shall implement Role-Based Access Control (RBAC) with four roles: admin (full access), developer (read/write development), viewer (read-only), and service (limited programmatic).	MUST
NFR-SEC-018	The MCP server shall implement capability-based access with granular capabilities: `tools:execute`, `resources:read`, `prompts:get`.	MUST
NFR-SEC-019	The MCP server shall enforce deny-by-default authorization: all access denied unless explicitly allowed.	MUST
NFR-SEC-020	The MCP server shall enforce proper role hierarchy inheritance.	MUST
NFR-SEC-021	The MCP server shall validate capability authorization on every protected endpoint request.	MUST

1.5.3.4 3.3.4 Security — Rate Limiting

ID	Requirement	Priority
NFR-SEC-022	The MCP server shall implement global rate limiting (default 1000 requests/minute).	SHOULD
NFR-SEC-023	The MCP server shall implement per-user rate limiting (default 60 requests/minute).	SHOULD
NFR-SEC-024	The MCP server shall implement per-API-key rate limiting (default 120 requests/minute).	SHOULD
NFR-SEC-025	The MCP server shall implement per-endpoint rate limiting with configurable limits.	SHOULD
NFR-SEC-026	The MCP server shall use the token bucket algorithm with burst support for rate limiting.	SHOULD
NFR-SEC-027	The MCP server shall include rate limit headers in responses: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`.	SHOULD
NFR-SEC-028	When rate limits are exceeded, the MCP server shall return 429 Too Many Requests with retry-after guidance.	SHOULD
NFR-SEC-029	Rate limits shall be configurable per deployment without code changes.	SHOULD

1.5.3.5 3.3.5 Security — Input Validation

ID	Requirement	Priority
NFR-SEC-030	The MCP server shall validate all inputs to prevent SQL injection using parameterized queries only.	MUST
NFR-SEC-031	The MCP server shall prevent command injection by never executing shell commands with user input; subprocess with argument lists only.	MUST
NFR-SEC-032	The MCP server shall prevent path traversal by validating paths against allowed base directories.	MUST
NFR-SEC-033	The MCP server shall prevent XSS by sanitizing output and HTML-encoding user content.	MUST
NFR-SEC-034	The MCP server shall prevent XXE by disabling external entities in XML parsing.	MUST
NFR-SEC-035	The MCP server shall prevent SSRF by validating and whitelisting external URLs.	MUST
NFR-SEC-036	The MCP server shall prevent ReDoS by limiting regex complexity and using timeouts.	MUST
NFR-SEC-037	The MCP server shall enforce a maximum request size of 1MB (configurable).	MUST
NFR-SEC-038	The MCP server shall enforce a maximum JSON nesting depth of 5 levels.	MUST
NFR-SEC-039	The MCP server shall enforce a maximum string length of 10,000 characters per field.	MUST
NFR-SEC-040	The MCP server shall enforce a request timeout of 30 seconds (configurable).	MUST

1.5.3.6 3.3.6 Security — Data Protection

ID	Requirement	Priority
NFR-SEC-041	The MCP server shall sanitize sensitive data (PII, secrets, tokens) in all log output.	MUST
NFR-SEC-042	The MCP server shall mask passwords (never log, hash with bcrypt/argon2), API keys (show as `sk_***abc123`), tokens (redact fully), and email addresses (show as `u***r@example.com`).	MUST
NFR-SEC-043	The MCP server shall enforce TLS 1.2 or higher for all network communication.	MUST

ID	Requirement	Priority
NFR-SEC-044	The MCP server shall store no hardcoded secrets; all credentials via external secrets management.	MUST
NFR-SEC-045	The MCP server shall maintain an audit trail for 100% of access attempts.	MUST

1.5.3.7 3.3.7 Security — Audit Logging

ID	Requirement	Priority
NFR-SEC-046	The MCP server shall log all security-relevant events: authentication attempts, authorization decisions, tool executions, resource access, configuration changes, rate limit violations, and error events.	MUST
NFR-SEC-047	The MCP server shall structure audit events as JSON with: timestamp, event_type, user_id, user_role, action, result, ip_address, correlation_id, and duration_ms.	MUST
NFR-SEC-048	The MCP server shall maintain audit logs as append-only and tamper-evident.	MUST
NFR-SEC-049	The MCP server shall include request correlation IDs in all audit entries.	MUST
NFR-SEC-050	The MCP server shall retain audit logs for a minimum of 1 year (configurable).	MUST

1.5.3.8 3.3.8 Security — Security Headers

ID	Requirement	Priority
NFR-SEC-051	The MCP server shall include `Content-Security-Policy: default-src 'self'` in all HTTP responses.	MUST
NFR-SEC-052	The MCP server shall include `Strict-Transport-Security: max-age=31536000; includeSubDomains` in all HTTP responses.	MUST
NFR-SEC-053	The MCP server shall include `X-Content-Type-Options: nosniff` in all HTTP responses.	MUST
NFR-SEC-054	The MCP server shall include `X-Frame-Options: DENY` in all HTTP responses.	MUST

ID	Requirement	Priority
NFR-SEC-055	The MCP server shall include `X-XSS-Protection: 1; mode=block` in all HTTP responses.	MUST
NFR-SEC-056	The MCP server shall include `Referrer-Policy: no-referrer` in all HTTP responses.	MUST
NFR-SEC-057	While operating in production, the MCP server shall restrict CORS origins to an allowed list with no wildcards.	MUST

1.5.3.9 3.3.9 Security — Provider Credential Management

ID	Requirement	Priority
NFR-SEC-058	The MCP server shall store all LLM provider credentials in an encrypted secret management service (AWS Secrets Manager, HashiCorp Vault, Azure Key Vault).	MUST
NFR-SEC-059	The MCP server shall never store API keys, tokens, or credentials in plain-text configuration files or environment variables.	MUST
NFR-SEC-060	The MCP server shall use secret references (e.g., `KV_AI_PROXY_KEY`) instead of direct credential values.	MUST
NFR-SEC-061	The MCP server shall support automatic credential rotation without service restart or code changes.	SHOULD
NFR-SEC-062	The MCP server shall restrict secret access to authorized services and users only.	MUST
NFR-SEC-063	The MCP server shall log all secret access attempts (success/failure) with service identity.	MUST
NFR-SEC-064	When using an enterprise gateway, the system shall inject mandatory headers (`X-Project-ID`, `X-Cost-Center`, `X-Environment`, `X-Request-ID`) that cannot be overridden by client requests.	SHOULD
NFR-SEC-065	When required enterprise headers are missing, the system shall reject the request with 401 Unauthorized.	SHOULD

1.5.3.10 3.3.9a Security — MCP Protocol Security Best Practices

ID	Requirement	Priority
NFR-SEC-073	The MCP server shall mitigate confused deputy attacks by requiring per-client user consent before executing tools or accessing resources on behalf of a user.	MUST
NFR-SEC-074	The MCP server shall not pass through user access tokens directly to downstream services (token passthrough is explicitly prohibited per MCP security best practices).	MUST
NFR-SEC-075	The MCP server shall enforce SSRF protections: validate all outbound URLs against an allowlist, block requests to private/internal IP ranges (10.x, 172.16–31.x, 192.168.x, 127.x, ::1), enforce HTTPS for remote resources, and validate redirect targets.	MUST
NFR-SEC-076	The MCP server shall prevent session hijacking by generating cryptographically secure, non-deterministic session identifiers and binding sessions to the authenticated user.	MUST
NFR-SEC-077	The MCP server shall not rely on session identity alone for authorization; sessions shall supplement, not replace, proper authentication and authorization checks.	MUST
NFR-SEC-078	When operating as a local server, the MCP server shall require user consent before executing any tool or exposing resources, and shall support sandboxing of tool execution environments.	MUST
NFR-SEC-079	The MCP server shall implement scope minimization: request only minimal OAuth scopes initially and use incremental scope elevation when additional permissions are needed.	MUST
NFR-SEC-080	The MCP server shall validate the `state` parameter and `redirect_uri` in OAuth flows to prevent CSRF and open redirect attacks.	MUST
NFR-SEC-081	When using consent cookies, the MCP server shall protect them with `Secure`, `HttpOnly`, `SameSite=Strict` attributes and bind them to specific tool/resource/argument combinations.	MUST

1.5.3.11 3.3.10 Performance

ID	Requirement	Priority
NFR-PERF-001	The MCP server shall respond to resource listing within 200ms (p95).	MUST
NFR-PERF-002	The MCP server shall respond to resource reading within 500ms (p95).	MUST
NFR-PERF-003	The MCP server shall complete typical tool execution within 2 seconds (p95).	MUST
NFR-PERF-004	The MCP server shall add less than 50ms protocol overhead per request.	MUST
NFR-PERF-005	The MCP server shall complete authentication within 100ms with cached JWKS.	MUST
NFR-PERF-006	When using the AI Service Provider Gateway, the gateway shall add less than 30ms overhead (p95).	SHOULD
NFR-PERF-007	The MCP server shall support 100 or more concurrent connections per instance.	MUST
NFR-PERF-008	The MCP server shall support resource sets of 10,000+ items.	MUST
NFR-PERF-009	The MCP server shall maintain a memory baseline of less than 500MB per instance.	MUST
NFR-PERF-010	The MCP server shall use a stateless design enabling linear horizontal scaling.	MUST
NFR-PERF-011	The MCP server shall support a throughput of more than 100 requests/second.	MUST
NFR-PERF-012	The MCP server shall maintain p50 response time below 100ms.	MUST
NFR-PERF-013	The MCP server shall maintain p99 response time below 1000ms.	MUST
NFR-PERF-014	The MCP server shall maintain an error rate below 0.1%.	MUST

1.5.3.12 3.3.11 Reliability

ID	Requirement	Priority
NFR-PERF-015	The MCP server shall degrade gracefully when dependencies are unavailable, continuing to serve available capabilities.	MUST
NFR-PERF-016	The MCP server shall automatically reconnect for transient failures.	MUST
NFR-PERF-017	The MCP server shall implement the circuit breaker pattern: open after 5 consecutive failures, half-open after 30 seconds.	MUST
NFR-PERF-018	The MCP server shall report clear health status via health check endpoints.	MUST

ID	Requirement	Priority
NFR-PERF-019	When the primary AI provider returns a 429 error, the gateway shall failover within 100ms.	SHOULD
NFR-PERF-020	When the primary AI provider returns a 5xx error, the gateway shall retry with exponential backoff (1s, 2s, 4s, 8s) then failover within 500ms total.	SHOULD
NFR-PERF-021	When the primary AI provider times out, the gateway shall immediately failover within 50ms.	SHOULD
NFR-PERF-022	When connectivity is lost, the circuit breaker shall detect failure within 200ms.	SHOULD
NFR-PERF-023	The MCP server shall maintain 99.9% availability.	MUST

1.5.3.13 3.3.12 Observability — Logging

ID	Requirement	Priority
NFR-OBS-001	The MCP server shall use structured JSON logging with configurable levels (ERROR, WARN, INFO, DEBUG).	MUST
NFR-OBS-002	The MCP server shall log requests and responses with sensitive data sanitized.	MUST
NFR-OBS-003	The MCP server shall include correlation IDs and context in all error logs.	MUST
NFR-OBS-004	The MCP server shall include performance metrics (duration, status) in log output.	MUST

1.5.3.14 3.3.13 Observability — Metrics

ID	Requirement	Priority
NFR-OBS-005	The MCP server shall expose a Prometheus-compatible metrics endpoint at `/metrics`.	SHOULD
NFR-OBS-006	The MCP server shall expose: `mcp_requests_total{endpoint, method, status}`, `mcp_request_duration_seconds{endpoint, quantile}`, `mcp_errors_total{endpoint, error_type}`, `mcp_active_connections`, `mcp_rate_limit_hits_total{tier}`.	SHOULD
NFR-OBS-007	The MCP server shall expose resource usage metrics (CPU, memory, connections).	SHOULD

1.5.3.15 3.3.14 Observability — Health Checks

ID	Requirement	Priority
NFR-OBS-008	The MCP server shall expose a `/health` liveness endpoint responding within 100ms.	MUST
NFR-OBS-009	The MCP server shall expose a `/ready` readiness endpoint (dependencies available) responding within 500ms.	MUST
NFR-OBS-010	The MCP server shall expose a `/startup` startup probe endpoint responding within 5 seconds.	MUST

1.5.3.16 3.3.15 Observability — Tracing

ID	Requirement	Priority
NFR-OBS-011	The MCP server shall support OpenTelemetry-compatible distributed tracing.	MAY
NFR-OBS-012	The MCP server shall propagate correlation IDs across requests.	MAY
NFR-OBS-013	The MCP server shall annotate spans for key operations.	MAY

1.5.3.17 3.3.16 Containerization — Docker Image

ID	Requirement	Priority
NFR-CNTR-001	The MCP server shall be packaged as a production-ready Docker container using a minimal, security-hardened base image (Alpine, distroless).	MUST
NFR-CNTR-002	The MCP server container image shall be less than 100MB compressed.	MUST
NFR-CNTR-003	The MCP server container shall support AMD64 and ARM64 architectures.	MUST
NFR-CNTR-004	The MCP server container shall have zero critical or high vulnerabilities in image scans.	MUST
NFR-CNTR-005	The MCP server container shall start and become healthy within 5 seconds.	MUST
NFR-CNTR-006	The MCP server container shall run as a non-root user (UID > 1000).	MUST
NFR-CNTR-007	The MCP server container shall use a read-only root filesystem where possible.	MUST

ID	Requirement	Priority
NFR-CNTR-008	The MCP server container shall drop all Linux capabilities and add only specific required ones.	MUST
NFR-CNTR-009	The MCP server container final image shall contain no shell or package managers.	MUST
NFR-CNTR-010	The MCP server container shall contain no secrets in image layers or environment variables.	MUST
NFR-CNTR-011	The MCP server container shall support automated rebuilds on base image updates.	MUST
NFR-CNTR-012	The MCP server container shall pass CIS Docker Benchmark checks.	MUST

1.5.3.18 3.3.17 Containerization — Configuration

ID	Requirement	Priority
NFR-CNTR-013	The MCP server shall support configuration via environment variables as primary source (12-factor app).	MUST
NFR-CNTR-014	The MCP server shall support configuration files mounted as volumes.	MUST
NFR-CNTR-015	The MCP server shall support secrets mounted from secret managers.	MUST
NFR-CNTR-016	The MCP server shall validate all configuration on startup with clear error messages.	MUST
NFR-CNTR-017	The MCP server shall support standard volume mount paths: `/config` (read-only), `/data` (read-write), `/secrets` (read-only), `/logs` (write).	MUST

1.5.3.19 3.3.18 Containerization — Orchestration

ID	Requirement	Priority
NFR-CNTR-018	The MCP server shall provide a Helm chart for Kubernetes deployment.	SHOULD
NFR-CNTR-019	The MCP server shall integrate with Kubernetes ConfigMaps and Secrets.	SHOULD
NFR-CNTR-020	The MCP server shall complete graceful shutdown (SIGTERM handling) within 30 seconds.	MUST
NFR-CNTR-021	The MCP server shall define appropriate resource requests and limits for Kubernetes.	SHOULD

ID	Requirement	Priority
NFR-CNTR-022	The MCP server shall be compatible with Kubernetes 1.24+.	SHOULD
NFR-CNTR-023	The MCP server shall provide a reference `docker-compose.yml` with common configurations.	SHOULD

1.5.3.20 3.3.19 Containerization — Distribution

ID	Requirement	Priority
NFR-CNTR-024	The MCP server container shall be published to GitHub Container Registry (ghcr.io).	MUST
NFR-CNTR-025	The MCP server container shall be published to Docker Hub.	MUST
NFR-CNTR-026	The MCP server shall be listed in the MCP Marketplace with complete metadata.	MUST
NFR-CNTR-027	The MCP server container shall implement image signing (Cosign or Docker Content Trust).	MUST
NFR-CNTR-028	The MCP server container shall include an SBOM (Software Bill of Materials).	MUST
NFR-CNTR-029	The MCP server container shall include provenance attestations.	MUST
NFR-CNTR-030	The MCP server container shall be published within 10 minutes of release via CI/CD.	MUST

1.5.4 3.4 Compliance Requirements

ID	Requirement	Priority
NFR-SEC-066	The MCP server shall support PII detection in data access patterns.	MUST
NFR-SEC-067	The MCP server shall support configurable masking of sensitive fields.	SHOULD
NFR-SEC-068	The MCP server shall support configurable data retention policies for logs and audit trails.	MUST
NFR-SEC-069	The MCP server shall support data subject deletion requests (right to erasure).	SHOULD

ID	Requirement	Priority
NFR-SEC-070	Where processing EU personal data, the MCP server shall support GDPR consent management, data portability, and audit trails.	SHOULD
NFR-SEC-071	Where processing California consumer data, the MCP server shall support CCPA privacy notices and opt-out mechanisms.	SHOULD
NFR-SEC-072	Where processing protected health information, the MCP server shall support HIPAA access controls, audit logs, and encryption.	SHOULD

1.6 4. Design Constraints

1.6.1 4.1 Protocol Constraints

ID	Constraint	Source
DC-001	The MCP server shall comply with MCP specification version 2025-11-25.	CP-01, CP-02
DC-002	The MCP server shall use JSON-RPC 2.0 as its message protocol.	MCP Spec
DC-003	The MCP server shall use JSON Schema 2020-12 for tool input validation.	MCP Spec (2025-11-25)
DC-004	The MCP server shall use OAuth 2.1 for HTTP transport authorization.	MCP Authorization Spec
DC-005	Production deployments shall use Streamable HTTP transport only (not stdio).	Security
DC-006	Production deployments shall be containerized.	Operations

1.6.2 4.2 Technology Constraints

ID	Constraint	Source
DC-007	The MCP server shall be implemented in Python using the FastMCP v3.x framework (ADR-001, ADR-002).	Architecture decision
DC-008	The MCP server shall use OCI-compliant container image format.	Operations
DC-009	The MCP server shall use structured JSON logging.	Observability

ID	Constraint	Source
DC-010	The MCP server shall use OpenTelemetry (OTEL) for metrics, traces, and logs.	Industry standard

1.6.3 4.3 Deployment Constraints

ID	Constraint	Source
DC-011	The MCP server shall follow 12-factor app principles for configuration.	Operations
DC-012	The MCP server shall implement stateless design for horizontal scaling.	NFR-PERF-010
DC-013	The MCP server shall support backward compatibility for at least 2 minor versions.	Operations
DC-014	The MCP server shall support rolling updates for zero-downtime upgrades.	Operations

1.6.4 4.4 Architecture Decision References

Key technical decisions are documented in Architecture Decision Records (ADRs) in the Architecture Description:

ADR	Decision	Relevant Requirements
ADR-001	FastMCP as MCP server framework	Enables rapid development
ADR-002	JWT/JWKS authentication method	NFR-SEC-010 through NFR-SEC-016
ADR-003	Stateless server design	NFR-PERF-010, DC-012
ADR-004	Database for tool metadata	Operational requirements
ADR-005	Streamable HTTP transport protocol	FR-PROTO-001, FR-PROTO-025–031, DC-005

1.7 5. Verification

Each requirement is mapped to a verification method per IEEE 29148:

Method	Description
T — Test	Verified through automated test execution
I — Inspection	Verified through code review or document review
A — Analysis	Verified through analysis of design or metrics
D — Demonstration	Verified through interactive demonstration

1.7.1 5.1 Functional Requirements Verification

Requirement	Method	Verification Description
FR-PROTO-001	T	Integration test: Streamable HTTP connection established, POST messages exchanged, SSE upgrade verified
FR-PROTO-002	T	Integration test: backward compatibility with deprecated HTTP+SSE transport verified
FR-PROTO-003	T	Integration test: stdio transport functional for local development
FR-PROTO-004	T	Contract test: all messages conform to JSON-RPC 2.0
FR-PROTO-005	T, A	TLS version verified in integration test; certificate analysis
FR-PROTO-006	T	Security test: invalid Origin returns 403
FR-PROTO-007	T	Unit test: stdout contains only JSON-RPC; stderr contains logs
FR-PROTO-008	T	Performance test: connection handshake < 2s
FR-PROTO-009	T	Unit test: malformed messages return structured error
FR-PROTO-010	T	Integration test: clean shutdown with no orphans
FR-PROTO-011	D, T	Demonstration with 3+ MCP clients (Claude Desktop, VS Code, Cursor)
FR-PROTO-012	T	Contract test: initialization response includes capabilities
FR-PROTO-013	T	Contract test: metadata includes name, version, description
FR-PROTO-014	T	Contract test: protocol version negotiation succeeds
FR-PROTO-015	T	Contract test: 100% JSON-RPC 2.0 compliance
FR-PROTO-016	I	Inspection: `server.json` contains complete metadata
FR-PROTO-017	I	Inspection: compliance with MCP moderation guidelines
FR-PROTO-018	T	Integration test: registry API schema compatibility
FR-PROTO-019	T	Unit test: all error codes match JSON-RPC 2.0 spec
FR-PROTO-020	T	Unit test: tool validation returns `isError: true`
FR-PROTO-021	T, I	Security test: no stack traces in responses; code review
FR-PROTO-022	T	Unit test: transient errors include retry guidance
FR-PROTO-023	T	Unit test: all errors include correlation ID
FR-PROTO-024	T	Unit test: error log entries contain full context
FR-PROTO-006a	T	Security test: local server bound exclusively to loopback interface

Requirement	Method	Verification Description
FR-PROTO-025	T	Contract test: `MCP-Session-Id` header returned during initialization and validated on subsequent requests
FR-PROTO-026	T	Integration test: invalid/expired session ID returns 404
FR-PROTO-027	T	Contract test: `MCP-Protocol-Version` header present in all responses after initialization
FR-PROTO-028	T	Integration test: SSE reconnection with `Last-Event-ID` redelivers missed events
FR-PROTO-029	T	Integration test: HTTP GET on MCP endpoint opens SSE stream for server-initiated messages
FR-PROTO-030	T	Integration test: per-request timeout enforcement and progress-based reset
FR-PROTO-031	T	Contract test: `ping` request receives immediate response
FR-PROTO-032	T	Integration test: server does not send requests before receiving `initialized` notification
FR-PROTO-033	T	Integration test: `notifications/cancelled` stops referenced request and frees resources
FR-PROTO-034	T	Unit test: `initialize` requests are never cancelled
FR-RSRC-001	T	Contract test: `resources/list` returns complete descriptors
FR-RSRC-002	T	Contract test: `resources/read` returns content by URI
FR-RSRC-003	T	Integration test: pagination with $>$ 100 resources
FR-RSRC-004	T	Unit test: text and base64 binary content
FR-RSRC-005	T	Unit test: invalid/nonexistent URI returns error code `-32002`
FR-RSRC-006	T	Performance test: 10MB+ resource via streaming
FR-RSRC-007	T	Contract test: `resources/templates/list` response
FR-RSRC-008	T	Unit test: RFC 6570 template expansion
FR-RSRC-009	D	Demonstration: parameter completion suggestions
FR-RSRC-010	T	Contract test: subscribe/unsubscribe endpoints
FR-RSRC-011	T	Integration test: notification within 5 seconds
FR-RSRC-012	T	Integration test: subscriptions cleaned on disconnect
FR-RSRC-013	T	Performance test: listing $<$ 200ms
FR-RSRC-014	T	Contract test: `title` and `size` fields present in resource definitions
FR-RSRC-015	T	Contract test: resource annotations (audience, priority, lastModified) validated
FR-RSRC-016	T	Integration test: `notifications/resources/list_changed` sent when resource set changes

Requirement	Method	Verification Description
FR-TOOL-001	T	Contract test: `tools/list` returns definitions with schemas
FR-TOOL-002	T	Unit test: schemas validate as JSON Schema 2020-12
FR-TOOL-003	T	Contract test: `tools/call` executes and returns results
FR-TOOL-004	T	Unit test: validation error returns `isError: true`
FR-TOOL-005	T	Unit test: invalid inputs rejected before execution
FR-TOOL-006	T	Unit test: text, binary, audio, and resource link result types
FR-TOOL-007	T	Load test: 5+ concurrent executions
FR-TOOL-008	T, I	Unit test: tool name 1–128 chars with valid characters; code review: naming conventions
FR-TOOL-009	I	Documentation review: descriptions and examples
FR-TOOL-010	T	Unit test: `icons` array (URI + media type) in tool/resource/prompt definitions
FR-TOOL-011	T	Integration test: User-Agent header present
FR-TOOL-012	T	Integration test: timeout enforced
FR-TOOL-013	T	Security test: no internal details in error messages
FR-TOOL-014	T	Integration test: exponential backoff on failure
FR-TOOL-015	T	Integration test: circuit breaker opens on failures
FR-TOOL-016	T	Integration test: progress notifications for long operations
FR-TOOL-017	T	Integration test: `notifications/cancelled` cancellation within 2 seconds
FR-TOOL-018	T	Configuration test: per-tool timeouts
FR-TOOL-019	D	Demonstration: tools visible in client UI
FR-TOOL-020	T	Integration test: audit log entries for all executions
FR-TOOL-021	T	Unit test: consent level metadata in tool definitions
FR-TOOL-022	T	Integration test: per-user enable/disable
FR-TOOL-023	T	Integration test: `notifications/tools/list_changed` sent when tool set changes
FR-TOOL-024	T	Contract test: `outputSchema` and `structuredContent` in tool results match JSON Schema 2020-12
FR-TOOL-025	T	Unit test: tool annotations (readOnlyHint, destructiveHint, idempotentHint, openWorldHint) present and valid
FR-PROMPT-001	T	Contract test: `prompts/list` returns templates
FR-PROMPT-002	T	Contract test: `prompts/get` with argument interpolation
FR-PROMPT-003	T	Integration test: embedded resource resolution
FR-PROMPT-004	T	Unit test: default values for missing optional args
FR-PROMPT-005	D	Demonstration: parameter completion
FR-PROMPT-006	I	Documentation review: descriptions and examples

Requirement	Method	Verification Description
FR-PROMPT-007	T	Contract test: optional `title` and `icons` fields in prompt definitions
FR-PROMPT-008	T	Unit test: audio content (data + mimeType) in prompt messages
FR-PROMPT-009	T	Unit test: invalid prompt name/missing args returns `-32602`
FR-PROMPT-010	T	Integration test: `notifications/prompts/list_changed` sent when prompt set changes
FR-SAMP-001–005	T	Contract test: sampling request/response flow
FR-ELIC-001–005	T	Contract test: elicitation create and response
FR-TASK-001–012	T	Integration test: full task lifecycle (creation, polling, result retrieval, cancellation, TTL expiry, access control, audit logging)
FR-ORCH-001–005	T, D	Multi-server integration test and demonstration
FR-GWWY-001–012	T	Gateway integration test suite (see UAT-GW groups)

1.7.2 5.2 Non-Functional Requirements Verification

Requirement	Method	Verification Description
NFR-SEC-001–009	T	OAuth 2.1 auth flow integration tests
NFR-SEC-010–016	T	JWT validation unit and integration tests
NFR-SEC-017–021	T	RBAC and capability authorization tests
NFR-SEC-022–029	T	Rate limiting integration tests
NFR-SEC-030–040	T	Input validation and injection prevention tests
NFR-SEC-041–045	T, I	Log sanitization tests; code inspection
NFR-SEC-046–050	T	Audit logging integration tests
NFR-SEC-051–057	T	HTTP response header tests
NFR-SEC-058–065	T, I	Secret management tests; configuration inspection
NFR-SEC-073–081	T	MCP protocol security best practices tests: confused deputy, token passthrough, SSRF, session hijacking, scope minimization, consent cookies
NFR-PERF-001–014	T	Performance and load tests
NFR-PERF-015–023	T	Reliability and failover tests
NFR-OBS-001–013	T, I	Observability integration tests; configuration inspection
NFR-CNTR-001–012	T, A	Container scanning, CIS benchmark, image analysis
NFR-CNTR-013–017	T	Configuration integration tests
NFR-CNTR-018–023	T, D	Kubernetes deployment tests; demonstration
NFR-CNTR-024–030	T, I	Registry publishing verification; CI/CD pipeline inspection

1.7.3 5.3 Test Coverage Targets

Test Type	Scope	Coverage Target
Unit	Functions, classes, modules	> 80%
Integration	API endpoints, data flows	> 70%
Contract	MCP protocol compliance	100%
Security	Auth, validation, injection	100% critical paths
Performance	Load, stress, latency	Key endpoints
End-to-End	User workflows	Critical paths

Component	Minimum Coverage
Tools	90%
Business Logic	85%
API Endpoints	80%
Utilities	80%

1.7.4 5.4 Gateway Test Groups

The AI Service Provider Gateway shall be verified through 6 test groups:

Group	Focus	Test IDs
1	Connectivity & Discovery	UAT-GW-1.1 through 1.4
2	Security & Governance	UAT-GW-2.1 through 2.5
3	Model Parity & Routing	UAT-GW-3.1 through 3.4
4	Fallback & Resiliency	UAT-GW-4.1 through 4.6
5	Performance & Latency	UAT-GW-5.1 through 5.3
6	MCP Sampling Integration	UAT-GW-6.1 through 6.3

1.8 6. Traceability Matrix

1.8.1 6.1 Requirements to Architecture Traceability

SRS Requirement	Architecture Section	Viewpoint (IEEE 42010)
FR-PROTO-001–010, 006a	01-architecture-overview.md	Functional
FR-PROTO-011–015	01-architecture-overview.md	Functional
FR-PROTO-016–018	07-deployment-patterns.md	Deployment
FR-PROTO-019–024	03-tool-implementation.md	Functional
FR-PROTO-025–034	01-architecture-overview.md	Functional
FR-RSRC-001–016	03b-resource-implementation.md	Functional

SRS Requirement	Architecture Section	Viewpoint (IEEE 42010)
FR-TOOL-001–025	03-tool-implementation.md	Functional
FR-PROMPT-001–010	03a-prompt-implementation.md	Functional
FR-SAMP-001–005	03c-sampling-patterns.md	Functional
FR-ELIC-001–005	03f-elicitation-patterns.md	Functional
FR-TASK-001–012	03g-task-patterns.md	Functional
FR-ORCH-001–005	03h-multi-server-orchestration.md	Functional
FR-GWWY-001–012	03i-ai-service-provider-gateway.md	Deployment
NFR-SEC-001–009	02-security-architecture.md	Security
NFR-SEC-010–016	02-security-architecture.md	Security
NFR-SEC-017–021	02-security-architecture.md	Security
NFR-SEC-022–029	02-security-architecture.md	Security
NFR-SEC-030–040	02-security-architecture.md	Security
NFR-SEC-041–045	02-security-architecture.md, 02a-data-privacy-compliance.md	Security, Information
NFR-SEC-046–050	05-observability.md	Operational
NFR-SEC-051–057	02-security-architecture.md	Security
NFR-SEC-058–065	03i-ai-service-provider-gateway.md	Security, Deployment
NFR-SEC-066–072	02a-data-privacy-compliance.md	Information
NFR-SEC-073–081	02-security-architecture.md	Security
NFR-PERF-001–014	06a-performance-scalability.md	Operational
NFR-PERF-015–023	06a-performance-scalability.md, 03e-integration-patterns.md	Operational
NFR-OBS-001–004	05-observability.md	Operational
NFR-OBS-005–007	05-observability.md	Operational
NFR-OBS-008–010	05-observability.md	Operational
NFR-OBS-011–013	05-observability.md	Operational
NFR-CNTR-001–012	07-deployment-patterns.md	Deployment
NFR-CNTR-013–017	07-deployment-patterns.md, 06-development-lifecycle.md	Deployment
NFR-CNTR-018–023	07-deployment-patterns.md	Deployment
NFR-CNTR-024–030	07-deployment-patterns.md	Deployment

1.8.2 6.2 Requirements to Test Category Traceability

SRS Requirement	Test Category	CI Phase
FR-PROTO-*	Contract, Integration	PR, Main
FR-RSRC-*	Contract, Unit, Integration	PR, Main
FR-TOOL-*	Unit, Contract, Integration	Pre-commit, PR, Main
FR-PROMPT-*	Unit, Contract	Pre-commit, PR

SRS Requirement	Test Category	CI Phase
FR-SAMP-*	Contract, Integration	PR, Main
FR-ELIC-*	Contract, Integration	PR, Main
FR-TASK-*	Integration	Main
FR-ORCH-*	E2E, Integration	Main, Release
FR-GWWY-*	Integration (UAT-GW groups)	Main, Release
NFR-SEC-001–009	Security, Integration	PR, Main
NFR-SEC-010–016	Security, Unit	Pre-commit, PR
NFR-SEC-017–021	Security, Integration	PR, Main
NFR-SEC-022–029	Integration, Load	Main
NFR-SEC-030–040	Security, Unit	Pre-commit, PR
NFR-SEC-041–057	Security, Integration, Inspection	PR, Main
NFR-SEC-058–065	Integration, Inspection	Main
NFR-SEC-073–081	Security, Integration	PR, Main
NFR-PERF-*	Performance, Load	Main, Release
NFR-OBS-*	Integration, Inspection	PR, Main
NFR-CNTR-*	Container, Security Scan	Main, Release

1.8.3 6.3 Core Principle Traceability

Principle	Enforcing Requirements
CP-01 Client Portability	FR-PROTO-011, FR-PROTO-014, FR-PROTO-025–031, DC-001
CP-02 Registry Distribution	FR-PROTO-016–018, NFR-CNTR-024–030
CP-03 Provider Agnostic	FR-GWWY-001–012, NFR-SEC-058–065
CP-04 Separation of Concerns	FR-ORCH-005, FR-TOOL-008

1.8.4 6.4 ADR Traceability

ADR	Requirements Supported
ADR-001 (FastMCP Framework)	FR-TOOL-001–018, FR-RSRC-001–013, FR-PROMPT-001–006
ADR-002 (JWT/JWKS Authentication)	NFR-SEC-010–016
ADR-003 (Stateless Design)	NFR-PERF-010, DC-012
ADR-004 (Database Selection)	FR-RSRC-003, NFR-PERF-008
ADR-005 (HTTP/SSE Transport)	FR-PROTO-001, DC-005

1.9 Document Approval

Role	Name	Date
Author	Mark Sigler	2026-02-23
Engineering Lead		
Security Lead		
Product Owner		