Secure Development Standards

Document	Secure Development Standards
Applies to	Any application developed under this standard. **MessageFoundry (MEFOR)** is the reference implementation (Appendix A).
Maintained by	Project maintainers (open-source). Each deploying organization assigns its own local owner.
Status	Released
Version	1.0
Date	June 12, 2026
License	Publishable under the project's open-source license; intended to be shared with adopters and reused across projects.
Review cadence	At least annually, and on any material architecture or threat change
Aligns to	NIST SP 800-218 (SSDF) · NIST SP 800-115 · NIST SP 800-53 · NIST SP 800-66 Rev. 2 (HIPAA Security Rule) · OWASP ASVS 5.0 Level 2

1. Purpose and scope

This is the secure development standard for an **open-source software project intended for use in regulated environments, including healthcare** (handling of protected health information, PHI).

It is written to serve three audiences:

1. **The development team** — a consistent engineering bar to build to, across this and future projects.
2. **Deploying organizations** — both the maintainer's own organization and any other organization that adopts the open-source software, who need evidence that it was built and tested securely.
3. **Future projects** — this standard is **project-agnostic**; any new application can be developed under it without rewriting it.

How the standard is structured. The body (§2–§9) states requirements that apply to *any* application built under this standard. Each application records its own specifics — technology stack, applicable verification scope, the interface mechanisms it implements — in a **per-project Applicability Profile**. MEFOR's profile is Appendix A; future projects add their own (Appendix B, C, …).

Open-source note. The software is developed in the open. This standard, and the project's security attestations, are publishable so that adopters can rely on them or extend them for their own environment.

2. Shared responsibility

Because the software is built by one party and deployed by others, responsibilities split cleanly. Stating the split prevents either side from assuming the other has it covered.

The is responsible for	The is responsible for
Secure development practices (§4)	Its own environment, host, and network security
Secure-by-default configuration	Identity, credential, and key management in its environment
Security testing and attestation of the software (§5)	Backups, disaster recovery, and availability
Vulnerability response and disclosure (§7)	Its own compliance program — HIPAA Security Rule obligations, risk assessments of the deployment, and Business Associate Agreements where applicable
Documentation and evidence (§8)	Operational monitoring, patching, and incident response

No certification or agreement is conferred by the software itself. A deploying organization operates the software under *its own* programs; an attestation that the software was built securely is evidence for that organization's assessment, not a substitute for it.

---

3. How this maps to NIST (overview)

Three NIST publications cover three different questions. Together they form the standard:

Question	NIST publication	Section
How is the software built? (secure development process)	SP 800-218 (SSDF)	§4
How is it tested? (security testing methodology)	SP 800-115	§5
What controls does it implement? (security/privacy controls + HIPAA safeguards)	SP 800-53 / SP 800-66 Rev. 2	§6

These are complementary to the **OWASP ASVS 5.0 Level 2** assessment used for independent review: ASVS verifies that the built application is secure; SSDF attests to how it was built; 800-53/800-66 map the safeguards.

A note on claims and wording (read before publishing any claim)

NIST does **not** issue certificates for these frameworks. The project may **display alignment claims**, but words them honestly and backs them with evidence:

- **Use:** "Built to NIST SP 800-218 (SSDF)," "NIST SSDF–aligned," "tested per NIST SP 800-115," "controls mapped to NIST SP 800-53 / 800-66 Rev. 2," "HIPAA-compliant deployment supported."
- **Do not use:** "NIST certified" or any phrasing implying a certificate exists.
- **Back every claim** with the implemented practice and its evidence (this standard, test reports, the ASVS attestation, the applicability profile). A self-attestation is a formal, legally significant declaration — only make it if it is true.
- A third-party assessor may **validate** an attestation; that raises its weight but is still not a NIST certificate.

---

4. Secure software development — NIST SP 800-218 (SSDF)

SSDF organizes secure development into four practice groups: **Prepare the Organization (PO)**, **Protect the Software (PS)**, **Produce Well-Secured Software (PW)**, and **Respond to Vulnerabilities (RV)**. Requirements below apply to every project; project-specific tooling is recorded in the applicability profile.

4.1 Prepare the Organization (PO.1–PO.5)

- **Security requirements (PO.1).** Security requirements are documented (this standard plus each project's standing contract, e.g., a `CLAUDE.md` or equivalent) and treated as first-class alongside functional requirements.
- **Roles and responsibilities (PO.2).** Maintainer/reviewer roles are defined; security ownership is explicit; onboarding includes secure-development orientation.
- **Supporting toolchains (PO.3).** Source control; CI/CD with automated security checks (§5.2); dependency and secret scanning; pinned, integrity-verified dependencies.
- **Security check criteria (PO.4).** Pass/fail release gates are defined (§5.4); a release does not ship with unresolved high/critical findings.
- **Secure development environments (PO.5).** Disk-encrypted developer machines; **no real PHI in development or test** (synthetic or de-identified data only); least-privilege access to repositories and environments.

4.2 Protect the Software (PS.1–PS.3)

- **Protect code from unauthorized access/tampering (PS.1).** Branch protection, required reviews, least-privilege access; no direct commits to the main branch; signed commits where supported.
- **Verify release integrity (PS.2).** Releases are versioned and integrity-verifiable (checksums/signatures); a software bill of materials (SBOM) is generated per release.
- **Archive and protect releases (PS.3).** Each released version, its build inputs, and its SBOM are archived to support incident analysis and reproducibility.

4.3 Produce Well-Secured Software (PW.1–PW.9)

- **Secure design and threat modeling (PW.1–PW.2).** Each interface and component is threat-modeled; trust boundaries are identified and documented; designs are reviewed against the security requirements before build.
- **Reuse well-secured components (PW.4).** Prefer vetted libraries; avoid rolling custom cryptography.
- **Secure coding practices (PW.5).** Mandatory:
- **Input validation** — validate structure and content at every ingress; reject or quarantine malformed input rather than processing it. *(Example: HL7 message validation in MEFOR.)*
- **Parameterized queries only** — no string-built SQL; use an ORM or parameterized statements throughout.
- **Authentication & authorization** — enforce on every action; deny by default; interface mechanisms per §6.4.
- **Web-service interfaces (REST and SOAP)** — authenticate every endpoint; validate and size-limit payloads against a schema; for SOAP/XML, disable external-entity resolution (XXE) and DTD processing; apply rate limiting and timeouts; never expose stack traces or sensitive data in fault responses.
- **File-handler interfaces** — confine reads/writes to configured directories and canonicalize paths (reject `../` traversal and symlink escapes); validate file type and size by content, not extension; use atomic write-then-rename so partial files are never processed; least-privilege storage, never an executable or web-served path; never execute file contents; scan inbound files for malware where feasible; encrypt sensitive files at rest and securely delete after processing per retention.

- **Cryptography** — TLS for all network communication; encryption at rest for sensitive data; approved algorithms and libraries; use FIPS-validated crypto where a deployment requires it.
- **Secrets** — never in code, prompts, or commit history; sourced from environment/secret store; enforced by pre-commit and CI secret scanning.
- **Error handling and logging** — fail closed; never log secrets or sensitive data; produce a tamper-resistant, timestamped audit log.
- **Secure build configuration (PW.6).** Reproducible builds; security-relevant build/interpreter and dependency settings fixed in the pipeline.
- **Code review and analysis (PW.7).** Every change is peer-reviewed; static analysis (SAST) and software composition analysis (SCA) run in CI (§5.2).
- **Test executable code (PW.8).** A maintained automated test suite runs on every change; security test cases are included.
- **Secure defaults (PW.9).** Ships secure-by-default (TLS on, encryption on, least-privilege accounts, verbose audit logging); insecure options require explicit, documented opt-in.

4.4 Respond to Vulnerabilities (RV.1–RV.3)

- **Identify on an ongoing basis (RV.1).** Continuous dependency monitoring; a defined intake channel for internally and externally reported issues (§7).
- **Assess, prioritize, remediate (RV.2).** Findings are triaged by severity with target remediation timelines (set per project in the profile); fixes are verified before closure.
- **Root-cause analysis (RV.3).** Significant vulnerabilities receive a root-cause review; systemic causes feed back into this standard.

---

5. Security testing and assessment — NIST SP 800-115

Testing follows the methodology of NIST SP 800-115 (*Technical Guide to Information Security Testing and Assessment*): review techniques, target identification and analysis, and target vulnerability validation.

5.1 Testing tiers

Tier	What	Cadence
Automated (in CI/CD)	SAST, SCA/dependency scan, secret scanning, unit/integration security tests	Every commit / build
Dynamic	DAST / authenticated testing of the running app	Per release and periodically
Independent review	Source-code review + penetration test, **scoped as OWASP ASVS 5.0 Level 2** (§5.3), conducted per 800-115	Before a production release; after major change; periodically thereafter

5.2 Internal testing (continuous)

- SAST and SCA run automatically; builds fail on new high/critical findings.
- Secret scanning runs pre-commit and in CI; the full git history is kept clean of secrets, credentials, keys, and any sensitive data.
- Security-focused test cases (authn/authz, input validation, error handling) are part of the standard suite.

5.3 OWASP ASVS 5.0 Level 2 — scope

The independent review is scoped to **OWASP ASVS version 5.0.0** (released May 2025), **Level 2**:

- **Version-pinned citation.** Requirements are cited as `v5.0.0-<chapter>.<section>.<requirement>`; identifiers changed substantially from 4.0.x, so the version is always stated.
- **Scale and level model.** ~350 requirements across **17 chapters**. Levels are **cumulative** — L2 includes all L1. **L2 is the standard level for applications handling sensitive data**; L3 (defence-in-depth for the highest-assurance contexts) is not the default target, though a single high-risk component (e.g., authentication) may be held to L3 within the same engagement.
- **Access required for L2.** L2 is a white-box / hybrid review: the assessor needs source code, developer access, documentation, and an authenticated test instance running **synthetic, non-PHI** data.
- **Documented Security Decisions (new in 5.0).** Each chapter opens with a requirement to document *how* its controls are applied and *why*. This standard, the per-interface threat models (PW.1), and the secure-default baseline (PW.9) serve as that documentation. **Each project documents which chapters are in scope and records exclusions with justification** — see the applicability profile. (Documenting exclusions is itself an ASVS practice.)
- **5.0 modernizations to honor.** Cryptography (V11) reflects current guidance, including post-quantum considerations; authentication and password rules (V6) align with NIST SP 800-63; ASVS 5.0 scopes to **applications and APIs** (host/network infrastructure is the deployer's responsibility, §2).

The 17 chapters (v5.0.0): V1 Encoding and Sanitization · V2 Validation and Business Logic · V3 Web Frontend Security · V4 API and Web Service · V5 File Handling · V6 Authentication · V7 Session Management · V8 Authorization · V9 Self-contained Tokens · V10 OAuth and OIDC · V11 Cryptography · V12 Secure Communication · V13 Configuration · V14 Data Protection · V15 Secure Coding and Architecture · V16 Security Logging and Error Handling · V17 WebRTC.

Per-project chapter applicability (in scope / excluded with justification) is recorded in the applicability profile.

5.4 Release gates

A production release requires: passing automated checks, no unresolved high/critical findings, current independent-review status (or a documented risk acceptance), and updated evidence (§8).

---

6. Security/privacy controls and HIPAA safeguards — NIST SP 800-53 / SP 800-66 Rev. 2

For deployments that handle PHI, the software implements controls drawn from the NIST SP 800-53 catalog, mapped to the HIPAA Security Rule using NIST SP 800-66 Rev. 2 (*Implementing the HIPAA Security Rule: A Cybersecurity Resource Guide*). *(A non-PHI deployment may scope the HIPAA mapping out; the control families still apply.)*

6.1 Applied 800-53 control families

Family	Applied to the software
AC — Access Control	Least-privilege, role-based access; deny by default
AU — Audit & Accountability	Tamper-resistant, timestamped audit logging; no sensitive data in logs
IA — Identification & Authentication	Authenticated access enforced on every action; strong credential handling; interface mechanisms per §6.4
SC — System & Communications Protection	TLS in transit; encryption at rest; trust-boundary enforcement
SI — System & Information Integrity	Input validation; flaw remediation (RV); message/data integrity and durability
CM — Configuration Management	Version control, reviewed changes, secure-default configuration, SBOM
CP — Contingency Planning	Backup/restore and replay capability *(deployer operates DR in their environment, §2)*
RA — Risk Assessment	Vulnerability assessment by the project; deployment risk assessment by the deployer
SA — System & Services Acquisition	Secure development practices (§4) and vetted third-party components

6.2 HIPAA Security Rule safeguard mapping (via 800-66 Rev. 2)

HIPAA safeguard	Representative requirement	Software implementation	800-53 family
Administrative	Security management, risk analysis, workforce/access management	This standard; least-privilege access; *deployer's risk analysis*	RA, AC, SA
Physical	Facility and device controls	*Deployer's environment*	(Deployer)
Technical — Access Control	Unique user ID, authentication, automatic logoff	Authenticated, role-based access; session controls	AC, IA
Technical — Audit Controls	Record and examine activity	Tamper-resistant, timestamped audit log	AU
Technical — Integrity	Protect data from improper alteration/destruction	Input validation; durable, ordered processing	SI
Technical — Transmission Security	Protect data in transit	TLS for all sensitive transport	SC
(Addressable) Encryption	Encrypt sensitive data at rest and in transit	Encryption at rest and in transit	SC

6.3 Deployment risk assessment

Each deploying organization conducts its own HIPAA Security Risk Assessment of the deployment (mapped to 800-66 Rev. 2). The project supplies evidence (§8) to support it; it does not replace it.

6.4 Interface authentication standard

Integration software authenticates **systems, not people**, on its interfaces. Each connection uses the strongest mechanism the partner system supports, drawn from the hierarchy below; the mechanism, scope, and credential reference for every connection are recorded in its connection definition. (Maps to ASVS V6/V9/V10/V12; 800-53 IA/SC;

HIPAA person-or-entity authentication and transmission security.) *Which mechanisms a given project implements is recorded in its profile.*

Preferred — system-to-system:

- **Mutual TLS (mTLS).** Client-certificate authentication over **TLS 1.2+ (prefer 1.3)** with strong cipher suites; validate the full chain to a trusted CA, check revocation (OCSP/CRL), rotate certificates before expiry. Where tokens are also used, prefer **sender-constrained (mTLS-bound) access tokens** (ASVS 5.0 V10).
- **OAuth 2.0 client-credentials grant.** The default for machine-to-machine API auth. Prefer **asymmetric client authentication (** `private_key_jwt` **)** over shared secrets; issue short-lived, per-connection scoped tokens; validate issuer, audience, expiry, and scope on every request.
- **SMART on FHIR (Backend Services).** For any FHIR REST interface, authenticate using the SMART **Backend Services** profile — OAuth 2.0 client-credentials with a **signed JWT client assertion** and `system/` scopes; validate granted scopes against the requested operation.

Directory / enterprise integration (e.g., Active Directory):

- **Run under a least-privilege service account — preferably a group-Managed Service Account (gMSA)** on Windows/AD — so the password is auto-rotated and never stored in configuration.
- **Use Kerberos / Integrated Windows Authentication**; prefer Kerberos over NTLM (disable NTLM where feasible) with correct SPNs.
- **Authenticate to databases with integrated authentication** (the service account) rather than a stored database password, where supported.
- **Perform directory lookups over LDAPS (LDAP over TLS) only** — never cleartext LDAP; bind with a least-privilege account.
- **Map roles to directory security groups** for centralized RBAC; if human operators authenticate, **federate to the enterprise identity provider (AD FS / Entra ID) via OIDC or SAML** rather than a local user store.

Legacy / interoperability tier *(supported, least-preferred, documented per connection):* HTTP Basic over TLS, per-connection API keys, or SOAP **WS-Security** (UsernameToken or, preferably, X.509 certificate tokens with message-level signing). Always over TLS; credentials vaulted, scoped per connection, and rotated. Used only when a partner system cannot support a preferred mechanism, with the exception recorded.

Across all mechanisms: TLS everywhere (no cleartext sensitive transport); credentials and keys in a secret store, never in code or config; per-connection least privilege; and per-connection IP allowlisting / network segmentation as defense-in-depth.

---

7. Open-source project security

Because the software is developed in the open and adopted by others, the project also maintains:

- **Repository hygiene.** No secrets or sensitive data ever committed; the full history is scanned and kept clean. A clear `LICENSE`.
- **Coordinated vulnerability disclosure.** A published `SECURITY.md` with a private reporting channel and a disclosure timeline; reported issues feed the RV process (§4.4).
- **Signed, verifiable releases.** Release artifacts are signed and accompanied by an SBOM (PS.2), so adopters can verify provenance and integrity.

- **Contribution review.** All external contributions are security-reviewed before merge; signed commits / DCO required; maintainers gate merges; dependency provenance is checked.
- **Adopter guidance.** A deployment/hardening guide so adopters can stand the software up securely and meet their §2 responsibilities.

---

8. Evidence and attestation

The project maintains a current evidence set so any claim is backed:

- This **Secure Development Standards** document and each project's standing contract.
- **SSDF practice evidence** (toolchain configuration, review records, SBOMs, secure-default settings).
- **Test results** — CI security-scan history; the independent **OWASP ASVS 5.0 Level 2** report and re-test results.
- **Per-project applicability profile** (Appendix A and onward).
- A **claims register** recording each published claim, its wording, and the evidence behind it.

Attestation posture. The software is self-attested as NIST SSDF–aligned, tested per NIST SP 800-115, verified against OWASP ASVS 5.0 Level 2, and built to support HIPAA-compliant deployment (controls mapped to NIST SP 800-53 / 800-66 Rev. 2). Third-party validation of the SSDF attestation and the ASVS 5.0 Level 2 assessment raises the weight of these claims. **Attestations are published with releases** so adopters can rely on them; each adopter still performs its own deployment risk assessment (§6.3). None of these is a NIST certificate; displayable certificates (SOC 2, ISO 27001, HITRUST) are a separate, organization-level track.

---

9. References

- NIST SP 800-218, *Secure Software Development Framework (SSDF) v1.1*
- NIST SP 800-115, *Technical Guide to Information Security Testing and Assessment*
- NIST SP 800-53, *Security and Privacy Controls for Information Systems and Organizations*
- NIST SP 800-66 Rev. 2, *Implementing the HIPAA Security Rule: A Cybersecurity Resource Guide*
- OWASP Application Security Verification Standard (ASVS) v5.0.0 (May 2025), Level 2

Appendix A — Applicability Profile: MessageFoundry (MEFOR)

The first project under this standard. Future projects add their own profile (Appendix B, C, …) using the same headings.

A.1 Project summary

MessageFoundry (MEFOR) is an open-source **HL7 v2.x integration engine** — a candidate alternative to commercial engines (Corepoint, Mirth Connect, Rhapsody, Cloverleaf). It routes and transforms clinical messages between systems.

Technology stack: Python 3.11+, FastAPI/uvicorn, aiosqlite/SQLite (WAL), `python-hl7` / `hl7apy`, PySide6 (desktop UI), Windows/PowerShell deployment; MLLP/TLS transport; SQLCipher (or database-native) encryption at rest. Durable message store with FIFO/per-key ordering and dead-letter handling.

A.2 Interfaces and surfaces

- **HL7 v2.x over MLLP/TLS** (inbound/outbound).

- **REST and SOAP** web-service interfaces.
- **File-handler interface** (file-drop pickup / output).
- **PySide6 desktop client** (no web UI at this time).

A.3 OWASP ASVS 5.0 Level 2 — chapter applicability

#	Chapter (v5.0.0)	In scope	Notes
V1	Encoding and Sanitization	Yes	HL7 input validation, output encoding, parameterized SQL
V2	Validation and Business Logic	Yes	HL7 structural/content validation; routing/business-rule checks
V3	Web Frontend Security	**No**	No browser-delivered UI (PySide6 desktop + APIs); documented exclusion. Re-scope if a web/admin UI is added
V4	API and Web Service	Yes	**Core surface** — REST and SOAP: authn/authz per endpoint, schema/payload validation, XXE/DTD defenses, rate limiting
V5	File Handling	Yes	File-handler interface: path confinement, content validation, atomic write-then-rename, malware scan, encryption at rest
V6	Authentication	Yes	Per §6.4; align to NIST SP 800-63
V7	Session Management	Yes	API/UI session controls, timeout/logout
V8	Authorization	Yes	Role-based, least-privilege, deny-by-default
V9	Self-contained Tokens	Yes	OAuth access tokens (JWT): verify signature, issuer, audience, expiry; reject `alg:none`
V10	OAuth and OIDC	Yes	OAuth client-credentials; SMART on FHIR (Backend Services) for FHIR; sender-constrained tokens preferred
V11	Cryptography	Yes	At-rest encryption (SQLCipher / DB-native); approved algorithms; post-quantum awareness
V12	Secure Communication	Yes	TLS / MLLP-over-TLS; mTLS for system-to-system
V13	Configuration	Yes	Secure-by-default; secrets management; SBOM
V14	Data Protection	Yes	PHI minimization, encryption, no PHI in logs
V15	Secure Coding and Architecture	Yes	Threat modeling, secure design, vetted components
V16	Security Logging and Error Handling	Yes	Tamper-resistant audit log; fail-closed errors; no PHI/secrets in logs
V17	WebRTC	**No**	Not applicable — no WebRTC; documented exclusion

In scope: 14 chapters (V1, V2, V4–V16). Documented exclusions: V3, V17.

A.4 Interface authentication mechanisms implemented

- **Preferred:** mTLS; OAuth 2.0 client-credentials; **SMART on FHIR (Backend Services)** for FHIR interfaces.
- **Active Directory:** gMSA service account; Kerberos/Integrated Windows Auth; database integrated authentication; LDAPS; role-to-group mapping; AD FS / Entra ID federation for human operators.

- **Legacy tier (per connection, documented):** Basic-over-TLS, API keys, SOAP WS-Security (X.509 preferred).

A.5 Project-specific items to confirm

- **Remediation SLA windows** (RV.2) — proposed: Critical ≤ 7 days, High ≤ 30 days, Medium ≤ 90 days. *Confirm.*
- **Applicable 800-53 control set** — confirm the tailored baseline with the deploying organization's security lead.

Version history

Version	Date	Change
0.1–0.5	June 12, 2026	MEFOR-specific drafts: NIST SSDF/800-115/800-53/800-66 mapping; OWASP ASVS 5.0 L2 scope; file-handler and interface-authentication standards
1.0	June 12, 2026	**Genericized** into a reusable, project-agnostic standard. Added shared-responsibility (§2) and open-source project security (§7). Moved MEFOR-specific scope and choices to **Appendix A** applicability profile. Supersedes the MEFOR-specific drafts.