



IAF SDR | SOFTWARE & DIGITAL | FIRST EDITION

SOFTWARE LIFECYCLE & EVIDENCE MANUAL

Secure development, digital tool validity, controlled releases, reliable services and continuing support.

PREPARED FOR IAF ADOPTION

Version 1.0 | 16 September 2026

Software Developers & Digital Tools
Corporate Recognition | English master text

International Automotive Federation

iaf.com

Document control

Item	Controlled value
Programme	IAF Software and Digital Systems Recognition Programme (IAF SDR)
Document	Software Lifecycle & Evidence Manual
Edition / version	First Edition Version 1.0
Prepared	16 September 2026
Status	Prepared for IAF adoption. Effective operation requires recorded adoption, validated pilots and an operational verification route.
Owner	International Automotive Federation
Master language	English
Public recognition wording	IAF Corporate Recognition - bounded to the approved legal entity, sites and activities.
Hierarchy	Standard including internal directives and normative annexes; adopted implementation notice within those rules; Status and Mark Rules; Audit and Software Lifecycle Manuals; Applicant Guide; controlled forms.
Authority boundary	Corporate implementation assurance. Corporate scope is separate from release approval; no blanket product certification, output endorsement or vehicle approval is conferred.
Control	Retain exact scope and criterion versions. Protect completed records, personal information and software evidence.

Purpose and use

Secure development, digital tool validity, controlled releases, reliable services and continuing support.

Shall denotes a requirement; should guidance; may permission. Examples are illustrative unless a normative provision expressly establishes a rule. No checklist score can compensate for failed safety, integrity, competence or an unsupported scope.

Contents

Document control	2
TM01 Build a bounded digital assurance case	4
TM02 Define families, hubs and actual events	4
TM03 Establish consequence and release authority	5
TM04 Connect requirements to release evidence	5
TM05 Design robust interfaces and degraded behaviour	5
TM06 Protect repositories, automation and secrets	6
TM07 Control components, rights and supply-chain change	6
TM08 Prove source-to-artifact integrity	7
TM09 Validate with adverse evidence	7
TM10 Establish confidence in engineering and test tools	7
TM11 Control safe release and deployment	8
TM12 Recover safely from updates and compromise	8
TM13 Handle vulnerabilities and incidents proportionately	9
TM14 Protect vehicle and user data	9
TM15 Validate AI and action-taking software	10
TM16 Operate reliable multi-tenant services	10
TM17 Keep support and exit commitments credible	10
TM18 Calculate assessment effort and coverage	11
TM19 Measure quality without rewarding concealment	11
TM20 Prepare future technology and international operation	12

TM01 Build a bounded digital assurance case

Implementation and evidence

A digital assurance case connects an exact organisational claim to implemented controls and actual outcomes. Begin with legal holder, lifecycle responsibility, hub, family, supported configurations and evidence period. Distinguish code development from integration, distribution and service operation; each role has different control and evidence obligations.

For each requirement link owner, implemented practice, original evidence, independent check and limitation. Triangulate source records, technical demonstration, staff reasoning and customer outcomes. Preserve failures and contradictory facts. A repository, policy collection or favourable dashboard is not sufficient evidence that the complete lifecycle is controlled.

Verification and practical application

Example: a diagnostic application reads vehicle data but also offers coding functions. Read-only competence cannot establish safe actuation or configuration capability. Identify separate authority, vehicle conditions, access controls and validation for consequential functions. If these are unsupported, exclude them visibly rather than using a broad software category.

Use the scope schedule for boundaries, self-assessment for implementation, lifecycle dossier for actual releases and checklist for assessment conclusions. The independent corporate decision does not endorse every tool output. A customer still needs the release evidence and integration acceptance appropriate to its actual use.

TM02 Define families, hubs and actual events

Implementation and evidence

Group a family by materially common architecture, lifecycle controls, competence and consequence, not merely a shared brand. Record release lines, configurations, deployment modes, intended use and support boundaries. A hosted booking service and embedded braking controller should not be pooled to manufacture a common history.

Identify every responsible development, integration or operating hub and its contribution. For initial/new scope each family/responsible hub needs six months of implemented controls and three real completed release/change events, plus internal audit and review. A governance-only centre demonstrates real oversight cases.

Verification and practical application

A release to 1,000 vehicles is one event for its family/hub, not 1,000 events. A controlled service configuration change can count if it has genuine operational purpose, relevant assurance and actual deployment. Repeated pipeline executions, demonstrations and invented trivial edits do not count as independent completed events.

A shared release may support more than one hub only where each hub's actual contribution and controls are verified. At surveillance, a stable supported product need not receive artificial changes; review existing events and current support/operating evidence. Document low volume honestly and do not waive the initial evidence floor.

TM03 Establish consequence and release authority

Implementation and evidence

Trace direct and indirect effects of failure: vehicle control, diagnostic false passes, incorrect design output, privacy loss, fleet disruption and significant financial decisions. A desktop tool can be consequential even when it never connects to a moving vehicle. Identify foreseeable misuse and integration assumptions.

Define competent owners for safety interfaces, security, data and release decisions, with stop authority and deputies. Separate consequential creation from independent verification/release challenge. Small firms may use external reviewers with suitable competence, controlled access and ordinary outcome-independent fees.

Verification and practical application

Example: a test tool incorrectly treats a timeout as a pass. The risk concerns every acceptance decision made through that path, not just a user-interface defect. Contain use, identify affected results and require verified correction before claiming the gate is restored. An executive acceptance of the defect is not sufficient.

Risk scores organise attention but cannot offset mandatory gates or known unacceptable conditions. Preserve reasoned nonblocking limitations and responsible acceptance, while prohibiting uncontained serious-harm or uncontrolled critical-exploitation conditions. Programme finding deadlines are not technical permission to continue unsafe operation.

TM04 Connect requirements to release evidence

Implementation and evidence

Write requirements with observable acceptance for behaviour, timing, accuracy, security, privacy, availability, accessibility and failure response as relevant. Record assumptions and excluded operating conditions. Distinguish a contractual aspiration from a verified technical requirement.

Maintain bidirectional links among requirement, design, source/configuration, review, test, defect, release and target state. Trace material defects back to affected requirements, versions and users. Store original evidence with stable identities rather than relying solely on a changing dashboard URL.

Verification and practical application

Example: a fleet command must only affect the authorised vehicle. Positive tests show a valid command works; negative tests must show a valid user cannot command another account's vehicle. Link both tests and actual deployment configuration to the authorisation requirement.

When a requirement changes, assess interfaces, variants, training, customer commitments and regression. If criteria are revised after a failed test, preserve the original failure and independent reason for the revision. Disabling the test or deleting the requirement does not establish that the underlying risk disappeared.

TM05 Design robust interfaces and degraded behaviour

Implementation and evidence

Document trust boundaries, privileges, shared resources, timing, units, schemas and state transitions. Define behaviour on missing, stale, malformed or adversarial data, overload and dependency loss. Separate authentication of identity from authority for a specific action.

Validate degraded states in the actual application context. A cloud service can remain reachable while returning wrong vehicle status; an embedded controller can reboot successfully while the

physical system is in an unsafe state. Define correctness and safe behaviour rather than treating connectivity as success.

Verification and practical application

Example: a charging application displays an old available connector state after losing the back-end connection. The interface should distinguish stale or unknown information and avoid silently charging for an unconfirmed service. Reconcile interrupted transactions and actual charger state before declaring completion.

Control interface changes and validate both syntactic and semantic compatibility. An import may parse correctly while reversing a unit, coordinate convention or time basis. Preserve integration assumptions and customer-side checks so isolated component success is not misrepresented as complete-system assurance.

TM06 Protect repositories, automation and secrets

Implementation and evidence

Use individual identities, appropriate strong authentication, least privilege and reviewed administrative access. Separate development, verification and production according to consequence. Record who can change source, pipeline definitions, release references, signing policy and deployed configuration.

Prevent untrusted contributions from receiving production secrets or altering trusted build evidence. Review runner images, caches, administrative paths and automation permissions. A protected main branch is ineffective if an unrestricted pipeline or administrator can replace the output afterward.

Verification and practical application

Example: a credential is removed from the latest source file but remains in repository history and logs. Assess exposure, revoke or rotate appropriately, review access and identify affected systems. Deleting one visible occurrence is not evidence that the secret is no longer usable.

Test emergency revocation and departure handling, including contractors and support accounts. Preserve authorised access evidence without collecting unnecessary personal activity. Source review for IAF may use a protected session; it need not transfer complete repositories or secrets to the assessor.

TM07 Control components, rights and supply-chain change

Implementation and evidence

Create a release-linked component inventory identifying component, version, origin, dependency relationship and relevant rights. Include material containers, firmware, libraries, models and datasets. Mark unknowns and scope limits honestly; a list of direct libraries may omit important transitive or runtime dependencies.

Assess critical suppliers for technical quality, security, support, change notice and access to needed evidence. Reconcile received artifacts against approved components and rights. Publicly accessible code or data is not automatically available for every commercial, redistribution or model-training purpose.

Verification and practical application

Example: an upstream package is replaced with an incompatible patch under the same internal dependency label. Identify which builds consumed it, verify actual versions and impacts, and prevent uncontrolled propagation. Do not assume an upstream recommendation validates all downstream vehicle configurations.

When a dependency becomes unsupported, evaluate reachability, consequence, replacement, compensating controls and support promises. Set dated action and escalation. The applicant

remains accountable for the interfaces it controls even when it cannot compel an external maintainer to provide a fix.

TM08 Prove source-to-artifact integrity

Implementation and evidence

Link approved source, dependencies, toolchain, build settings, identity and outputs in a protected provenance record. Preserve the actual artifact and verification needed to recognise it later. A checksum proves that two byte sequences match; it does not prove that the source was authorised or the behaviour is correct.

Test reproducibility where feasible. Where timestamps, proprietary tools or other nondeterminism prevent byte-for-byte identity, document the cause and a controlled equivalence check capable of detecting consequential differences. Do not label an untested rebuild reproducible.

Verification and practical application

Example: source review passes, but the distributed binary came from a developer workstation with an unrecorded library. The missing trust link requires investigation and a controlled rebuild or other sufficient evidence. Signing that binary afterward does not repair the provenance gap.

Protect release/signing authority and test key compromise, revocation and recovery. Verify trust identity and target compatibility at distribution boundaries. A cryptographically valid package can still be malicious, obsolete or intended for a different hardware configuration.

TM09 Validate with adverse evidence

Implementation and evidence

Plan representative unit, integration, system and operational-context verification, justified by consequence and diversity. Include boundary values, resource exhaustion, bad data, timing, concurrency and dependency failure where relevant. Test volume and code coverage are supporting measures, not complete proof of fitness.

Preserve original inputs, environment, versions, timestamps, failures, reruns and defect disposition. Distinguish changed tests from repeated tests and explain omitted variants. Independent reviewers need access to source results and rationale, not just a selected success report.

Verification and practical application

Example: 99.8% of tests pass, but the failed tests concern cross-tenant authorisation and safe update recovery. The percentage cannot compensate for these blockers. Resolve the failure or establish a technically coherent excluded scope; do not average essential gates against cosmetic successes.

Regression must follow change impact, including configuration, data, infrastructure and model changes. Validate that a fix addresses the cause without introducing new failure modes. Preserve unsuccessful attempts and investigate shared failures across families rather than declaring every issue isolated.

TM10 Establish confidence in engineering and test tools

Implementation and evidence

Identify whether the tool produces, transforms or checks evidence used in a consequential decision, and whether errors can be detected independently downstream. Validate the actual intended use and configuration. A vendor demonstration or broad tool reputation cannot establish confidence for every engineering application.

Use independently known cases, known failures, boundary values, wrong units, stale data and missing inputs. Check end-to-end result and export integrity. A diagnostic timeout, missing sensor channel or incomplete test must not silently become a valid pass.

Verification and practical application

For a chosen conservative upper-limit rule, let measured result be y , expanded uncertainty U and limit L . Pass if $y + U$ is at most L ; fail if $y - U$ exceeds L ; otherwise inconclusive. With $L = 10$ and $U = 0.2$, values 9.7, 10.1 and 10.3 yield pass, inconclusive and fail. This is an illustrative agreed rule, not a universal acceptance rule.

If a unit conversion or solver error invalidates prior outputs, identify every affected tool version and result population, assess decisions and notify entitled users. Correcting the current executable does not retrospectively validate earlier reports or downstream designs.

TM11 Control safe release and deployment

Implementation and evidence

The release package should identify intended use, requirements, exact configuration, component inventory, original validation, independent approvals, known nonblocking limitations, support and recovery. Preserve the actual decision and authorised artifact. Commercial acceptance is separate from technical release approval.

Define target eligibility, safe preconditions, staged rollout, observation window, stop thresholds and recovery before deployment. Verify the installed or activated state and relevant function, not just transfer acknowledgement. Reconcile offline, failed and partially updated targets.

Verification and practical application

Example: 1,000 targets are intended; 920 show validated activation, 40 only downloaded, 25 failed and 15 are unreachable. Confirmed activation is 92%, not 96%. The remaining 80 require their respective disposition; an unreachable target is not a successful deployment.

For emergency remediation retain essential authority, provenance, targeted validation, safety and recovery controls. Additional nonessential documentation can follow under a controlled exception, but no retrospective approval makes an uncontained serious condition conforming. Record what was actually known at release time.

TM12 Recover safely from updates and compromise

Implementation and evidence

Validate recovery against physical and data states. Rollback may be unsafe after irreversible state changes, schema migration or trust revocation. Define a tested rollback, roll-forward, isolation or supported service route appropriate to the actual failure; a generic rollback policy is insufficient.

Exercise interruption, low power where relevant, connectivity loss, wrong-target delivery, partial installation and corrupted metadata in safe representative environments. Coordinate customer and integration responsibilities. Preserve evidence of failed recovery attempts and the resulting limitations.

Verification and practical application

Example: a back-end schema migration makes older vehicle-client software incompatible. Reverting only the server binary may worsen the failure. Validate coordinated recovery or a compatible intermediate state and communicate required customer action before widening deployment.

If signing or distribution is compromised, identify affected trust paths, artifacts and targets, stop unsafe propagation, re-establish trusted release authority and validate clean recovery. Distinguish blocking hostile downgrade from permitting an explicitly validated recovery path. Preserve user protection during key transition.

TM13 Handle vulnerabilities and incidents proportionately

Implementation and evidence

Provide accessible intake and secure exchange for vulnerability reports. Record receipt, affected versions, reporter communication, evidence and authority for any testing. A reporting policy should encourage lawful good-faith disclosure without implying blanket permission to intrude into customer or vehicle systems.

Triage using actual exposure, reachability, exploitation evidence, available controls and consequences. Set risk-based dated containment and remediation commitments with owners and escalation. Do not impose a universal patch deadline that ignores safety validation, or use a long deadline to tolerate known serious uncontrolled risk.

Verification and practical application

Example: a dependency vulnerability exists in the inventory but may be unreachable in one configuration and exposed in another. Verify both conclusions and their assumptions. A temporary network control may reduce exposure but must be checked for bypass, expiry and impact; it is not automatic permanent closure.

Notify IAF within 48 hours of awareness when the serious-incident trigger applies, without postponing emergency or earlier duties. Preserve chronology and uncertainty. Programme Minor/Major closure clocks apply to recognition findings and are not permission to delay necessary technical containment or customer notice.

TM14 Protect vehicle and user data

Implementation and evidence

Map actual data subjects, purposes, roles, authority and recipients across development, telemetry, support and model use. Separate vehicle owner, driver, passenger, employee and fleet customer. A contractual relationship with one party does not necessarily grant rights over all associated personal data.

Minimise location, images, behaviour and identifiers; verify logs and diagnostic exports as well as primary databases. Control support impersonation and third-party access. Protect source and engineering secrets separately from personal information while retaining necessary technical context.

Verification and practical application

Example: a debugging export contains precise trip history and access tokens although the fault can be investigated with minimised event codes. Redesign the export, revoke exposed secrets if necessary and assess prior recipients. A generic user consent statement does not make unnecessary data collection sound practice.

Implement retention, deletion, lawful holds and cross-border controls for actual duties. Release assurance evidence may need long retention, but raw personal telemetry does not inherit that period automatically. Explain backup and derived-data treatment and verify that deletion or de-identification preserves essential lawful evidence.

TM15 Validate AI and action-taking software

Implementation and evidence

Inventory model purpose, version, provider, data, consequence and operating domain. Separate code assistance, human decision support and autonomous action. Assign a competent responsible owner and define what the system may never do without verified authority and appropriate oversight.

Preserve training/evaluation lineage and rights, independence of evaluation, subgroup/rare-event limits and adverse results. Test misleading inputs, out-of-domain conditions and untrusted retrieved instructions. A benchmark score cannot establish suitability for a different vehicle population or consequential task.

Verification and practical application

Example: a maintenance assistant reads a service document containing an instruction to export credentials. The retrieved text is evidence to interpret, not authority to override access controls. Test bounded tool permissions and require appropriate human approval for consequential actions; record actual actions and rejections.

Monitor drift, provider updates and incorrect outputs. Revalidate material changes, keep a suitable fallback and correct affected users. Verify generated code, facts and citations against authentic sources and label synthetic exhibits. Neither an AI confidence score nor a polished explanation replaces technical evidence.

TM16 Operate reliable multi-tenant services

Implementation and evidence

Define indicators at the user-relevant boundary: availability, correct data, latency distribution, command success and recovery. State measurement windows, exclusions and dependencies. A responding endpoint returning stale or incorrect results is not necessarily a successful service.

Test tenant isolation, negative authorisation, rate limits, support access and administrative change. Validate regional failure, shared dependency loss, overload and state recovery safely. Production configuration, feature flags and infrastructure definitions require change control when they alter behaviour.

Verification and practical application

Example: a service has 43,200 eligible minutes in a 30-day month and 60 minutes of counted outage. Availability is $(43,200 - 60) / 43,200 = 99.8611\%$. Explain exclusions and separately report incorrect-data incidents; the availability figure cannot establish functional accuracy.

Exercise recovery with reconciliation of actual data, target inventories, permissions and evidence links. Restored infrastructure without correct state is incomplete recovery. Define safe degradation and customer communication when the service supports long-lived vehicles or consequential operations.

TM17 Keep support and exit commitments credible

Implementation and evidence

Record supported release lines, configurations, security maintenance, dependencies and planned support end before commitment. Resource the promised period and review actual customer/vehicle consequences. The programme deliberately sets no universal product-support duration; the organisation must justify and fulfil its actual obligations.

Assess supplier exit, acquired codebases, discontinued platforms and staff loss. Maintain usable customer data export and migration, necessary evidence custody and safe alternatives. Source escrow can help a specific contract but is not automatically usable without rights, dependencies, build knowledge and competent people.

Verification and practical application

Example: a cloud subscription ends while vehicles still rely on it for a material function. Explain this dependency upfront and plan lawful safe transition rather than silently disabling functionality. Review security and safety implications of continued offline use and customer responsibilities.

Retain necessary release/assurance evidence ten years after last delivery or declared support end, whichever is later, subject to binding limits and holds. Minimise personal information and preserve technical provenance. Recognition expiry does not terminate ongoing support, correction or contractual duties.

TM18 Calculate assessment effort and coverage

Implementation and evidence

Initial/renewal minimum is two central auditor-days plus one per hub up to ten technical personnel, two for 11-50 and three above 50. Add one hub-day for each material factor: novel consequential software/AI, continuous multiregion operations, complex critical outsourcing and unresolved systemic risk. An auditor-day is eight actual assessment hours excluding travel.

Example: an eight-person remote hub and a 25-person physical test hub with novel consequential software and complex critical outsourcing require at least $2 + 1 + 2 + 1 + 1 = 7$ auditor-days. Add time for adequate family demonstrations and evidence; central and hub hours cannot be credited twice.

Verification and practical application

Annual demonstrations include all consequential, new/materially changed and adverse-case families, deduplicated, plus $\text{ceil}(\sqrt{N})$ of the rest. With four mandatory families and 17 remaining, the floor is $4 + 5 = 9$ families. Rotate so every family is demonstrated within three years and expand where risk requires.

Three responsible hubs each supporting two families need at least $3 \times 2 \times 3 = 18$ event-sample selections. The same real event can support multiple hub contributions only when each is evidenced. All hubs are reviewed annually; physical technical visits and approved remote-only alternatives follow IR02. Numerical floors never cap necessary work.

TM19 Measure quality without rewarding concealment

Implementation and evidence

Define populations, periods, numerator, denominator, exclusions and source for escaped defects, failed changes, recovery, response and support performance. Preserve original and revised delivery commitments. Avoid targets that reward hiding defects, splitting changes artificially or removing difficult users from the denominator.

Example: six changes requiring rollback or immediate remediation among 120 production changes in a defined cohort give a 5% failed-change rate under that definition. State the observation window and count reopened incidents consistently. Compare like-for-like consequence and scope before claiming improvement.

Verification and practical application

Internal audit should examine original source-to-release traces, adverse cases, independence, suppliers, support and current operations at least annually. Management review should decide resources, priorities and corrective actions based on the full evidence, not merely throughput or favourable test counts.

Corrective action requires containment, evidence-based cause analysis, affected-release review and demonstrated subsequent effectiveness. Preserve failed attempts. A policy update or training session is an action, not proof that the original failure mechanism and downstream impact were controlled.

TM20 Prepare future technology and international operation

Implementation and evidence

Apply one consistent core while identifying actual local permissions, data duties, professional obligations and customer acceptance. Keep source evidence and translations traceable. Neither global branding nor IAF membership establishes universal legal approval or automatic acceptance of a software product.

For long-lived vehicles, inventory cryptographic trust, keys, sensitive data horizons, hardware limits and supplier support. Test migration, revocation, compatibility and safe recovery. A claim about future-resistant technology must be substantiated and must not distract from current access, signing and update weaknesses.

Verification and practical application

Introduce novel AI, digital twins, new computing or connected infrastructure through bounded pilots, representative/adverse validation, competent people and actual implementation before scope extension. Measure accessibility and resource impacts with a useful-work baseline; do not reduce essential verification merely to improve a compute metric.

Before programme launch complete four representative pilots including a failed or contested case. Audit operator controls annually and independently review at least every three years. Use real errors, complaints, cost/access barriers and decision consistency to improve the scheme without silently weakening essential gates.