Manage XDALC Manifesto Versions, Updates and Offline Access

Document type: Implementation guidance. **Framework reference:** XDALC-V001.

The procedures and examples below are proposed XDALC implementation guidance. They do not introduce a certification scheme or claim that a particular deployment has passed an evaluation.

An AI integration needs a stable reference for the principles it follows. If a live webpage changes silently, two sessions may apply different instructions while appearing to use the same framework. XDALC implementations should therefore distinguish the currently published material from the specific release adopted by a deployment.

This article proposes a release-management approach for XDALC-V001. File names, fields and workflows shown below are illustrative conventions. They do not claim that a download endpoint, signed release service or automated migration system already exists on xdalc.com.

Pin a complete policy bundle

Record the manifesto version together with the definitions and operational interpretations required by the application. Preserve a local copy of the reviewed bundle. A stable manifesto reference is insufficient if its definition of authorization can change independently without review.

Keep the bundle separate from the application-specific permissions. An update to wording should not silently grant access to a new tool. Record which version of the local instruction template interprets the release and which tests evaluated the resulting behavior.

```
jsonCopy

 {
   "example_schema": "xdalc-local-adoption-record-v1",
   "manifesto_version": "XDALC-V001",
   "bundle_revision": "operator-assigned-revision",
   "instruction_revision": "operator-assigned-prompt-revision",
   "update_mode": "review_before_activation",
   "offline_mode": "use_verified_approved_bundle",
   "integrity_digest": "REPLACE_WITH_DIGEST_OF_EXACT_BUNDLE_BYTES",
   "approved_by": "REPLACE_WITH_ACCOUNTABLE_OPERATOR",
   "approved_at": "REPLACE_WITH_APPROVAL_TIMESTAMP"
 }
```

The placeholders must be replaced before operational use. This example is a proposed record format, not a supported XDALC protocol. Store actual verification results and approval information outside model-generated assertions.

CC BY 4.0 License | HTML reference

Distinguish integrity from authenticity

A digest can show that bytes match an expected value. It cannot identify the publisher unless the expected value comes through a trusted channel. Downloading both a modified file and its matching digest from a compromised location does not establish authenticity.

If releases use signatures, operators must establish trusted keys, a rotation procedure and a response to compromise. The Update Framework provides a reference architecture using signed metadata, file hashes, version information and expiry to address update attacks. Applying such an approach to XDALC would require an implemented publishing and verification process; merely mentioning signatures is insufficient.

Choose exactly what is hashed. Hashing stored file bytes makes whitespace changes significant. If an implementation instead signs canonicalized JSON, use a defined scheme and a conforming library. RFC 8785 describes JSON canonicalization for repeatable cryptographic operations; it is an Informational RFC, not an Internet Standards Track specification.

Review behavioral changes before activation

A candidate update should state what changed, why it changed and which behaviors may be affected. Compare both wording and operational meaning. An apparently small change from “may” to “must” can matter more than a large editorial rewrite.

1. Obtain the candidate bundle through the established release channel.
2. Verify its origin and integrity according to the deployment's trust policy.
3. Compare the candidate with the adopted bundle and identify affected workflows.
4. Run relevant behavioral and enforcement tests in a staging environment.
5. Obtain the required operator approval and record the activation decision.
6. Activate the bundle consistently across the deployment and monitor affected behavior.

Never let a model infer that a larger version identifier automatically deserves authority. Discovery, verification, evaluation and adoption are separate events. An update feed can announce a release without granting permission to install it.

Define offline behavior by capability

If the site is unavailable, a deployment can continue using its approved local bundle when its operating policy permits. Record the bundle version and the most recent successful update check. Do not claim the copy is current when that cannot be verified.

The offline decision depends on the task. A cached manifesto may remain sufficient for drafting an itinerary, while unavailable current fare data prevents a reliable booking proposal. Separate unavailable policy updates from unavailable operational facts. Pause only the capabilities that lack the evidence or authorization they require.

Define review intervals and conditions that suspend sensitive operations, including expired trust metadata

where applicable. There is no universal safe offline duration for every application. Operators should choose a justified policy and document what happens when it is exceeded.

Prepare recovery without unsafe rollback

Keep the previous approved configuration and a migration record. A controlled rollback may help recover from a faulty update, but do not automatically return to a version withdrawn for a serious defect. Track rejected or revoked bundles so recovery does not reintroduce a known failure.

If no acceptable bundle is available, restrict affected capabilities and involve the responsible operator. Reverting a policy configuration also does not undo completed transactions; those require their own reconciliation and recovery procedures.

Make continuity visible

Each deployment should be able to state which release it follows, why that release was accepted and how its next update will be reviewed. Archives and verified local copies support the project's ambition to remain a durable reference, while acknowledging that no website can promise uninterrupted availability forever.