BONAFIDE™ SPECIFICATION V1.0 — PART 14

Institutional Branches, Leases & Policy

Lease-based institutional access, machine-executable branch policy, institutional transfer protocol, policy immutability, and user-consented retention

1. Purpose

This part defines the protocol for institutional branch lifecycle management: how institutions gain access to user vault data, what governs that access, how access changes over time, and what happens when a relationship ends or transfers to another institution.

The specification adds five protocol-level mechanisms: lease-based institutional access (replacing persistent keys), machine-executable branch policy (the governing contract for each branch), a transfer protocol for institutional changes, policy immutability (protecting users from unilateral term changes), and user-consented retention (ensuring the protocol never overrides the user's original consent).

This part does not define business logic, data taxonomies, or regulatory frameworks. It defines the protocol primitives that enable institutions and users to negotiate, enforce, and audit their data relationships.

2. Lease-Based Institutional Access

2.1 Principle

Institutions do not hold persistent decryption keys. Instead, institutions hold leases: time-bounded, auto-renewable decryption credentials that authorize access to specific vault quanta within a branch. A lease grants the institution the ability to decrypt data for a finite period. The lease renews automatically unless revoked by the user.

This inverts the current model where institutions receive permanent keys during peering. Leases ensure that institutional access is always finite, always auditable, and always revocable—while protecting institutions from operational disruption through automatic renewal.

2.2 Lease Properties

Property	Description
Term	Duration of the lease in days. Default: 30 days. Configurable per branch policy. Minimum: 1 day. Maximum: 365 days.
Auto-renewal	Lease renews automatically at expiration unless explicitly revoked. Protects institutions from access loss if the user is inactive or unreachable.
Scope	The set of quanta the lease authorizes access to, defined by branch, channel, and security level. A lease cannot exceed the branch policy's declared scope.
Revocability	The user can revoke a lease at any time. Revocation follows the branch policy's declared retention and grace period terms.
Cryptographic binding	The lease is bound to the institution's certificate, the branch identifier, and the

	lease term. A different institution, branch, or term produces a different lease key.
Ledger recording	Every lease event (grant, renewal, revocation, expiration) is recorded in the immutable ledger with cryptographic proof.

2.3 Lease Lifecycle

Grant

When a user peers with an institution, the protocol generates a lease scoped to the new branch. The lease key is derived from the peering handshake material, the institution's certificate, and the lease term. The lease is recorded in the ledger.

Renewal

At the lease term boundary, the protocol automatically generates a new lease key and re-wraps the branch's DEKs. The old lease key is invalidated. The institution's operational access is uninterrupted. The renewal is recorded in the ledger.

Revocation

The user initiates revocation through their vault interface. The protocol consults the branch policy to determine what happens to each data category (immediate purge, downgrade, or retention under the policy's declared terms). The revocation is recorded in the ledger.

Expiration

If auto-renewal is disabled (by user configuration or policy), the lease expires at the term boundary. The institution's lease key becomes non-functional. DEK wrappings for the institutional key path are removed. The expiration is recorded in the ledger.

2.4 Lease and the Validator Network

The validator network validates lease lifecycle events—not individual data operations. Validators confirm lease grants, renewals, revocations, and expirations. They do not validate every read or write within an active lease. This keeps validator traffic proportional to relationship events (thousands per day per institution), not transaction volume (potentially billions).

Day-to-day operations within an active lease are governed by the local ledger. The local ledger syncs lease state with the global ledger on lease events, not on every transaction.

3. Branch Policy

3.1 What a Branch Policy Is

Every institutional branch must publish a cryptographically signed policy document recorded on the ledger. The branch policy is the governing contract for the institutional relationship. It declares what data the institution stores, at what security levels, for how long, and what happens under various lifecycle events (revocation, transfer, account closure).

The policy is machine-executable: it consists of typed attributes with a defined schema that can be evaluated by software or hardware enforcement engines. The policy is also human-readable: each attribute has an associated description rendered in the user's vault interface.

3.2 Constitutional Minimums

The specification defines a small set of inviolable rules that no branch policy can override. These are the protocol-enforced constitutional minimums:

- **Notification:** The institution must notify the user of any policy change through the Bonafide protocol.
- **Consent:** The user must explicitly accept any policy change. Continued use does not constitute consent.
- **Transfer consent:** The user must explicitly accept institutional transfers. Silence defaults to rejection.
- **Finite leases:** Leases must have finite terms with auto-renewal. No perpetual grants.
- **Revocability:** The user can always revoke a lease, subject to the retention terms they originally accepted.
- **Transparency:** The institution must declare what data it retains and for how long before the relationship begins.
- **Proof:** Cryptographic proof of every lease event (grant, renew, revoke, transfer) is recorded on the ledger.
- **No retroactive weakening:** A policy update cannot reduce protections on data already stored under the previous policy without user acceptance.

3.3 Policy Attribute Schema

The policy consists of typed key-value attributes organized by namespace. The specification defines the attribute schema (field names, data types, valid value ranges, and operator set). Institutions fill in the values. The schema is extensible—institutions and industry groups can add namespaces—but cannot remove or weaken the constitutional attributes.

Core Attribute Namespaces

Namespace	Scope	Example Attributes
lease.*	Lease terms and behavior	lease.term_days, lease.auto_renew, lease.on_expire, lease.grace_period_days
data.*	Data classification and	data.<category>.security_level,

	handling	data.<category>.user_visible, data.<category>.on_revoke, data.<category>.retention_days
transfer.*	Transfer behavior	transfer.require_user_accept, transfer.default_on_silence, transfer.notice_days
retention.*	Post-closure retention	retention.<category>.duration_days, retention.<category>.security_level, retention.<category>.legal_basis

Attribute Value Types

Type	Format	Operators	Example
integer	Whole number	==, !=, <, >, <=, >=	data.pii.security_level = 8
boolean	true / false	==, !=	lease.auto_renew = true
enum	Defined value set	==, !=, in	data.pii.on_revoke = purge
duration	Integer + unit (days, hours)	==, !=, <, >, <=, >=	lease.term_days = 30
string	UTF-8 text (descriptions only)	N/A (not evaluated)	data.pii.description = "Personal..."

Action Enumerations

The on_revoke and on_expire attributes use a defined action set:

Action	Behavior
purge	Remove institutional DEK wrappings immediately. Data becomes undecryptable to the institution.
purge(N)	Remove institutional DEK wrappings after N-day grace period.
retain	Maintain institutional access under the declared retention terms (duration, security level, legal basis).
downgrade(L)	Change security level to L. Higher security, more restricted access. Useful for archival.
hash	Replace plaintext with cryptographic hash. Institutional proof of data existence without content access.
redact	Replace with redacted summary (ghost quantum). Institution sees proof of category, not content.

3.4 Machine Execution

The policy attribute set compiles to a binary format loadable by the Bonafide runtime and, on Classification S deployments, by the HSM's Security Enforcer IP block. The binary format is a lookup table of attribute-value pairs that the enforcement engine evaluates as gate conditions: integer comparisons, boolean checks, and enum lookups. No loops, no function calls, no complex branching.

The specification defines the binary format. How institutions author the policy (YAML, XML, GUI, or any other tool) is an implementation concern.

3.5 Hierarchical Policy Resolution

Policies can inherit from parent policies using a hierarchical namespace path. A base policy defines defaults. Sector-specific and jurisdiction-specific policies override specific attributes. The final resolved attribute set is the policy that governs the branch.

Resolution order: most specific wins. Constitutional minimums always override. Conflicts between same-level policies are rejected—the institution must resolve ambiguity before publication.

The specification defines the resolution algorithm. The Foundation does not maintain the hierarchy—institutions and industry groups maintain their own policy trees.

4. Institutional Transfer Protocol

4.1 When Transfers Occur

Institutional transfers occur when a user's branch data changes hands: acquisitions, mergers, divestitures, regulatory receivership, or voluntary institution-to-institution migration. The Bonafide protocol does not allow automatic inheritance of branch data. Every transfer requires explicit user action.

4.2 Transfer Flow

Step 1 — Transfer Intent

Institution A publishes a transfer intent to the ledger: the target institution (B), the effective date, and Institution B's published branch policy. This is a protocol-level event recorded on the ledger.

Step 2 — Policy Comparison

The Bonafide protocol compares Institution A's policy (the one the user agreed to) against Institution B's policy. Differences are categorized: equivalent, improvement, or material change. Jurisdiction changes are flagged as critical events regardless of other attributes.

Step 3 — User Notification

The user receives a protocol-level notification describing the transfer and the policy comparison. The notification includes: the transferring and receiving institutions, a summary of policy differences, and the user's options.

Step 4 — User Decision

The user has two options:

- **Accept:** Data transfers to Institution B under Institution B's policy. A new lease is established with Institution B. Institution A's lease is revoked.
- **Reject:** The branch is closed under Institution A's original policy terms. Revocation and retention rules from the original policy govern. The user's private vault copy (if synced) becomes the canonical copy.

Step 5 — Default on Silence

If the user does not respond within the notice period (declared in the branch policy, default: 60 days), the default action is reject. Silence is not consent. The branch is closed under the original policy terms.

4.3 Transfer and Lease Mechanics

A transfer is a lease revocation on Institution A followed by a lease grant to Institution B. The user's key wrappings are unaffected. Institution A's DEK wrappings are removed. Institution B's DEK wrappings are created from the new lease. The transfer event is recorded in the ledger.

5. Policy Immutability

5.1 Principle

An institution cannot unilaterally change the branch policy the user agreed to. If an institution wants to update their retention terms, change their data classification, or alter any policy attribute, they must:

- Publish a new policy version to the ledger (cryptographically signed, timestamped).
- Notify the user through the Bonafide protocol.
- Receive the user's explicit acceptance.

If the user rejects the update, the original policy remains in force for that user's branch. The institution may offer the new policy to new users. Existing users retain their original terms until they explicitly accept new ones.

5.2 Policy Versioning

Every policy version is recorded on the ledger with a cryptographic hash, the publishing institution's signature, and a timestamp. The ledger provides an immutable history of every policy the institution has published and every version each user has accepted.

The user's vault interface shows which policy version governs each branch and alerts the user to pending policy update requests.

5.3 Grandfathering

Multiple policy versions can coexist for the same institution. User A accepted policy v1. User B accepted policy v3. Both are active simultaneously. The institution's runtime enforces the correct policy version per user. The HSM's Security Enforcer evaluates the correct attribute set per branch.

6. User-Consented Retention

6.1 Principle

The Bonafide protocol never overrides the user's consent. Whatever data an institution retains after lease revocation or account closure, the user agreed to when they accepted the branch policy. The protocol enforces what the user already consented to—nothing more.

There is no protocol-granted regulatory lease. There is no mechanism for the protocol itself to extend institutional access beyond what the user originally accepted. If an institution needs to retain data for regulatory compliance, the policy must declare this up front, and the user must accept it before the relationship begins.

6.2 Retention Mechanics

When a lease is revoked, the protocol evaluates the branch policy's retention attributes for each data category:

Policy Attribute	Example Value	Protocol Action
on_revoke = purge	Immediate	Institutional DEK wrappings removed. Institution's copy becomes dead ciphertext immediately.
on_revoke = purge(90)	90-day grace	Institutional access continues for 90 days (for account settlement, final statements, etc.), then wrappings removed.
on_revoke = retain	Regulatory	Institutional access continues for retention.duration_days at retention.security_level. When retention expires, wrappings removed automatically.
on_revoke = downgrade(12)	Archival	Security level elevated to 12. Access restricted to compliance-only operations. Lease continues under retention terms.

6.3 Retention Expiration

When a retention period expires, the protocol automatically removes institutional DEK wrappings from the affected quanta. The institution may retain the ciphertext indefinitely—it is meaningless bits without the key. The expiration event is recorded in the ledger.

The user's vault interface reflects the current state: which institutions hold active leases, which hold retention leases, and when each retention expires.

6.4 User Visibility

The branch policy declares which data categories are user-visible (data.<category>.user_visible = true). User-visible quanta can be synced to the user's private vault as read-only copies. The institution pushes user-visible quanta to the user's vault on a schedule or on demand. The user sees their PII, account summary, and statements in their own vault. The institution never opens an inbound channel. The user never reaches into the institution's infrastructure.

Data the institution creates for its own operational purposes (internal risk scores, compliance flags, process artifacts) is not user-visible and is not governed by the Bonafide protocol. That is the institution's own data.

7. Compliance Requirements

7.1 Implementation Requirements

A compliant Bonafide implementation must enforce the following:

- **Lease enforcement:** Institutional access must be lease-based with finite terms. Persistent institutional keys are non-compliant.
- **Policy evaluation:** The runtime must evaluate the branch policy's attribute set for every lease event and every access authorization. The runtime must reject operations that violate the policy.
- **Constitutional enforcement:** The runtime must reject any policy that violates the constitutional minimums (Section 3.2), regardless of what the institution publishes.
- **Ledger recording:** Every lease event and policy acceptance must be recorded in the immutable ledger with cryptographic proof.
- **Transfer protocol:** The runtime must implement the full transfer flow (Section 4.2) including default-on-silence rejection.
- **Retention enforcement:** The runtime must automatically remove DEK wrappings when retention periods expire.

7.2 Hardware Enforcement

On Classification S deployments, the branch policy's binary attribute set is loaded into the HSM's Security Enforcer IP block. The enforcer evaluates policy attributes as hardware gate conditions: integer comparisons, boolean checks, and enum lookups against FPGA block RAM lookup tables. Policy enforcement in silicon cannot be bypassed by host OS compromise.

The Bonafide HSM Reference Design (separate document) specifies the binary policy format, the loading protocol, and the formal verification requirements for the Security Enforcer block.

7.3 Software Enforcement

On Classification A, B, and C deployments, the Bonafide runtime enforces policies in software. The runtime evaluates the same attribute set using the same logic—the difference is the trust boundary. Software enforcement is correct but can be bypassed by compromising the host. Hardware enforcement is correct and tamper-resistant.

Bonafide™ — Privacy by architecture, not by promise.

An open specification by Sly Technologies Inc. | bonafide.id | bonafideid.org

V1.0 — February 2026