Pathix — Security Whitepaper

Version: 1.0
Publication Date: 2026-06-23
Classification: Public
Audience: Information Security, Compliance, Risk, and Procurement Teams.

Table of Contents

1. Executive Summary

Pathix is a forensics and visibility platform for Microsoft Dynamics 365 (D365) and Microsoft Dataverse environments. It answers two operational questions that are painfully hard today: "*what writes to this field?*" (dependency analysis across plugins, classic workflows, modern flows, business rules, form scripts, dataflows, canvas apps, and more) and "*who can do what, and what is the blast radius?*" (security-role analysis, shared-

role detection, privilege creep). It is sold as a single product, licensed per-tenant, annually, banded by active Dataverse principals.

This document describes the security architecture, controls, and design principles that govern the platform. It supports information-security reviews, third-party risk assessments, procurement evaluations, and internal compliance audits.

Pathix's security posture rests on four foundational design commitments:

1. **Customer-hosted deployment.** Pathix is delivered as an Azure Resource Manager (ARM) / Bicep template that the customer deploys into their own Microsoft Azure subscription. The application and all of its data execute exclusively within the customer's tenancy. Pathix L.L.C. — the vendor — maintains **no production hosting infrastructure that processes customer data**.
2. **Metadata-only data boundary.** Pathix is *architecturally constrained* to ingest, store, and retrieve metadata only. It has **no technical capability to access customer business record contents.** The constraint is enforced at the SDK-wrapper layer, the database-schema layer, and through code review. *Customer-authored source bodies* (plugin IL, workflow XAML, flow JSON, form JavaScript, FetchXML, PowerFx) are parsed in memory and **never persisted to the database** — end-to-end, no exceptions.
3. **Passwordless authentication.** Microsoft Entra ID is used for end-user authentication; a system-assigned Managed Identity is used for service-to-service authentication with Microsoft Dataverse. No long-lived secrets, client passwords, or API keys are required for the platform to operate.
4. **Bring Your Own Key (BYOK) for AI, off by default.** Optional AI features operate exclusively against an AI provider configured by the customer (Microsoft Azure OpenAI, Anthropic, or OpenAI). Pathix never proxies, brokers, or stores AI provider credentials. The deterministic core of the product works with zero AI.

Key security property. Because Pathix executes within the customer's Azure tenancy and is architecturally constrained to metadata, granting a user access to Pathix does **not** extend their access to any Dataverse record they could not otherwise view. The application's internal permission model does not need to mirror Dataverse data permissions.

Personal data — stated accurately, not hidden. "Metadata-only" does *not* mean "no personal data." To model the security graph, Pathix stores **identifiers for administrative principals** (user display names and User Principal Names, drawn from the `systemuser` table, plus team membership and role assignments). That is personal data. It resides in the customer's own Azure SQL Database, in the customer's tenant, under the customer's RBAC. The vendor never receives it. See §5 for the precise framing.

2. Product Overview

2.1 Purpose and Use Cases

Pathix addresses two challenges widely recognized in the Dynamics 365 / Dataverse administrative community:

- **Dependency analysis** — "*What components write to or read from this field?*" Pathix performs static analysis across plugin steps, classic workflows, business rules, dialogs, custom actions, modern Power Automate flows, form-bound JavaScript, PowerApps Component Framework (PCF) controls, canvas applications, formula columns, dataflows, ribbon customizations, saved queries, and form configurations.
- **Authorization and blast-radius analysis** — "*Who is authorized to perform this action, and what is the impact of role membership?*" Pathix analyzes the Dataverse security model — roles, privileges, principals, teams, business units, and field-level security profiles — and surfaces findings such as shared roles, orphaned plugin steps, field-secured columns without profile grants, and similar privilege anomalies.

2.2 Platform Components

Pathix is composed of the following logical components, all deployed within the customer's Azure subscription:

Component	Function
Web Application & API	ASP.NET Core (.NET 10) application hosting the user interface and HTTP API
Scan Orchestrator	Background service that performs metadata extraction and analysis
Relational Database	Azure SQL Database storing scan metadata, findings, and configuration
Secrets Store	Azure Key Vault holding the license token and any optional integration secrets
Identity	System-assigned Managed Identity used for authenticated access to Dataverse
Observability	Application Insights instance owned by the customer for logs and metrics
MCP Server	An in-product Model Context Protocol server exposing the dependency/security graph to MCP-aware AI agents (subject to license validity)

2.3 Licensing in Brief

Pathix licenses are issued by the vendor as **signed JSON Web Tokens (JWT)** that bind to the customer's Entra tenant identifier and carry a band claim. Validation is **air-gapped by default**: a local cryptographic signature check, no network call required. Online validation is optional plumbing for renewal hints only. A lapsed or invalid license **soft-degrades, never hard-bricks** — it blocks new scans and the MCP server but **never deletes, locks, or gates access to data already scanned.**

3. Deployment Model

3.1 Customer-Hosted Architecture

Pathix is delivered exclusively as a customer-hosted application. The vendor does not operate a multi-tenant Software-as-a-Service platform that processes customer data. Customers receive an ARM/Bicep template

that provisions the following resources within an Azure subscription of their choosing:

- Azure Container App (compute)
- Azure SQL Database (storage)
- Azure Key Vault (secrets)
- System-assigned Managed Identity (identity)
- Azure Application Insights (telemetry, customer-owned)
- Optional: Azure OpenAI binding for AI features (customer-configured)

3.2 Data Residency

All persistent data created by Pathix resides in the Azure region selected by the customer at deployment time. The vendor has no production database, file storage, or compute infrastructure that processes customer data. Data residency, retention, and sovereign-cloud commitments are inherited from the customer's Azure subscription configuration.

3.3 Tenant Isolation

Because each customer deploys an independent instance of Pathix into their own Azure subscription, tenant isolation is provided by Microsoft Azure's subscription boundary. There is no shared application, shared database, or shared compute layer between customers. A compromise of one customer's environment cannot, by architecture, expose another customer's data.

3.4 Air-Gapped and Restricted-Network Environments

Pathix supports operation in environments with **no outbound internet connectivity**. Because license validation is air-gap-by-default (Section 14.2), the product functions fully offline. Telemetry collection is opt-in and may be disabled entirely with no impact on functionality.

> Sovereign-cloud / region authorities (e.g., US Government, US DoD) beyond Microsoft commercial cloud are supported per-deployment, determined by the customer's Azure environment. Confirm specifics at engagement.

4. Data Handling and the Metadata-Only Boundary

4.1 Architectural Commitment

Pathix ingests, stores, and is technically capable of retrieving **metadata only.** This is the foundation of the rest of the security and compliance posture. It is enforced in code and schema, not policy alone.

4.2 What Pathix Reads, and What It Does Not

Category	Examples	Pathix access
Structural metadata	Table and column definitions, relationships	Read
Customization configuration	Workflow definitions, plugin registrations, modern flow definitions, business rules, form scripts, ribbon customizations, saved queries, canvas-app configurations	Read
Security configuration	Roles, privileges, principals (including user display names and UPNs), team memberships, field-level security profiles	Read — *see §5 for the personal-data framing*
Customer business records	Account, contact, opportunity, and custom-entity records	**No access**
Audit log values	Before / after payload values from record changes	**No access**
Embedded record values	Hardcoded record GUIDs in workflow XAML or formulas	**Discarded post-parse, never persisted**

4.3 No Customer-Authored Source Is Persisted

Customer-authored source bodies — plugin IL, workflow XAML, modern flow JSON, form JavaScript, FetchXML, PowerFx formulas — are parsed **in memory during a scan and then discarded.** Nothing customer-authored is persisted to the database. End-to-end, no exceptions. Derived structural outputs (the dependency edges, reference records, finding rows) are persisted; the bodies that produced them are not. If a parser improvement needs to re-examine a component, Pathix re-pulls from Dataverse rather than retaining a copy.

4.4 Enforcement Layers

The metadata-only boundary is enforced at three independent architectural layers:

Layer 1 — Dataverse SDK Wrapper (`Pathix.Dataverse`). All communication with Microsoft Dataverse is routed through a dedicated internal assembly, `Pathix.Dataverse`, which exposes the `IDataverseClient` interface and its implementation. The interface surface is deliberately constrained to metadata-shaped methods (`GetTableMetadata`, `GetSecurityRoles`, `GetWorkflowDefinitions`, `GetPrincipals`, etc.). It does not expose generic `Retrieve` or `RetrieveMultiple` methods, nor the underlying Dataverse organization service. Adding a method to the wrapper requires explicit code review against the metadata-only constraint.

The boundary is enforced by an **architectural fitness test** that runs in continuous integration on every pull request and every push to the main branch. The test (`DataverseSdkContainmentTests.OnlyAllowedAssembliesReferenceDataverseSdk`) enumerates every loaded `Pathix.*` assembly and fails the build if any of them — outside the wrapper and the test fixtures — references the Microsoft Dataverse SDK packages (`Microsoft.PowerPlatform.Dataverse.Client`, `Microsoft.Xrm.Sdk`, `Microsoft.Crm.Sdk.Proxy`). Enforcement is at the **assembly-reference / NuGet-package level** — stronger than a source-token scan, because the SDK types cannot be used without

referencing the package, and the package reference itself is what the test catches. A violation blocks merge to `main`.

Layer 2 — Database Schema Constraints. The Pathix relational schema contains no columns capable of holding customer record values. There is no generic `Value`, `Content`, or `Data` column. Customer-authored content is processed in memory during a scan and is not persisted, per §4.3.

Layer 3 — Code-Review Discipline. Every change affecting Dataverse ingestion is reviewed against an explicit checklist: *Does this change introduce a new path for record data to enter Pathix? Does it expose record values in logs, error messages, or telemetry? Does it introduce a UI element capable of displaying customer record contents?*

4.5 Customer-Side Verification

A security team can verify the metadata-only boundary independently:

1. **Audit the Pathix Scanner role** in your Power Platform Admin Center. Every privilege Pathix *requests* is a **Read** at Organization scope on one of the system tables Pathix declares. (Platform caveat: where SharePoint document-management integration is enabled, Microsoft auto-appends SharePoint table privileges — including write and create — to all custom roles, the Pathix Scanner role included; these are platform-injected, not requested by Pathix, and Pathix never exercises them. See §6.3 — they are expected, not a finding.)
2. **Capture network traffic from Pathix's Container App.** Outbound calls to Dataverse should be `RetrieveAllEntitiesRequest`, `WhoAmI`, and reads against the documented system tables. No `RetrieveMultiple` against business tables.
3. **Audit the Pathix SQL database.** The schema contains no `Value` / `Content` / `RecordData` columns.
4. **Run a Pathix scan with Dataverse audit logging enabled** and inspect the application user's call pattern in the Dataverse audit trail.

If anything in a network capture or audit trail does not match what Pathix declares, that is a bug — please disclose it via the channel in §10.2.

5. Personal Data (stated accurately)

> This section deliberately does not say "Pathix processes no personal data." It is the kind of claim a careful reviewer will probe, because it is not quite true and a Data Protection Officer is right to challenge it. Below is what is precisely true.

To model the security graph, Pathix stores identifiers for administrative **principals** in the customer's environment — principally:

- **User display names and User Principal Names (UPNs)** from the `systemuser` table.
- **Team membership and role assignments** for those principals.

That information is personal data under the GDPR / UK-GDPR and analogous regimes. The precise framing is as follows:

- **This personal data concerns administrators / principals in the security model**, not the customer's end-user or end-customer business records.
- **It is stored in the customer's own Azure SQL Database**, inside the customer's tenant, governed by the customer's Azure RBAC. **The vendor (Pathix L.L.C.) does not receive or hold it.**
- **The customer remains the controller** of this personal data. Retention, access, and any data-subject requests are exercised through the customer's own Azure controls. Pathix L.L.C.'s position is that of a software supplier, not a data processor, because it never receives the data.

On compliance frameworks attached to record content — HIPAA, PCI-DSS, GDPR data-subject rights *against record content* — these obligations do not attach to Pathix because Pathix holds no business record content. This is **not** a claim that no privacy law applies; the principal personal data above is personal data and remains in the customer's control.

6. Identity and Access Management

6.1 End-User Authentication

Pathix uses Microsoft Entra ID for end-user authentication via the Microsoft Authentication Library (MSAL), implementing OAuth 2.0 Authorization Code with Proof Key for Code Exchange (PKCE). Pathix does not maintain its own user credential store. Multi-factor authentication, conditional access, and identity protection policies configured at the customer's Entra ID tenant level apply transparently to Pathix authentication events.

6.2 Service-to-Service Authentication

Pathix authenticates to Microsoft Dataverse using a system-assigned Azure Managed Identity bound to the Azure Container App. **No client secrets, certificates, or connection strings containing credentials are stored in Pathix configuration.** Token acquisition uses Azure.Identity's `DefaultAzureCredential` chain.

6.3 Principle of Least Privilege

The Managed Identity is registered as an Application User in each Dataverse environment that Pathix scans, holding a single custom Dataverse security role — **Pathix Scanner** — packaged as a Managed Power Platform Solution. The Pathix Scanner role grants **read-only at Organization scope** on a fixed set of system tables required for metadata extraction. As Pathix defines it, the role requests no write, create, delete, append, or share privileges anywhere in the customer's environment.

Platform caveat: in any environment where SharePoint document-management integration is enabled, Microsoft automatically appends SharePoint table privileges — including read, write, and create — to **all** custom security roles, the Pathix Scanner role included, and re-adds them if they are removed. These privileges are injected by the platform, not requested by Pathix, and Pathix never exercises them: it reads metadata through the SDK and never accesses SharePoint document content.

The complete authoritative privilege list ships with the Pathix Scanner managed solution and may be audited by the customer in the Power Platform Admin Center *prior to assignment*. The high-level structure is: read privileges on the plugin family (`pluginassembly`, `plugintype`, `sdkmessage`, `sdkmessageprocessingstep`, `sdkmessageprocessingstepimage`); workflow / customization family (`workflow`, `webresource`, `systemform`, `savedquery`, `savedqueryvisualization`, `customization`); newer surfaces (`appaction`, `msdyn_dataflow`, `connectionreference`, `serviceendpoint` — covering service endpoints and webhooks, `canvasapp`, `publisher`, `customapi`); security family (`role`, `fieldsecurityprofile`, `team`, `position`, `solution`, `systemuser`, `organization`); and the metadata layer accessed via `RetrieveAllEntitiesRequest`. The exact, name-for-name privilege matrix — the form a reviewer sees in the Power Platform Admin Center — is reproduced in **Appendix A**.

6.4 Internal Authorization

Within the Pathix application, **Pathix 1.0 does not implement in-app role separation**: any user authenticated through the customer's Microsoft Entra tenant sees the full model. Access to the application is therefore governed by the customer's Entra access policies — the customer controls who may sign in. In-app role-based authorization (e.g., admin vs. viewer distinctions) is a documented post-1.0 layer. For a forensics tool deployed by and for the customer's own security and administration team, this is a conscious, documented posture.

6.5 Auditability of Provisioning

All Pathix internal authentication and configuration events are recorded as structured log records and emitted to the customer's Application Insights instance. Customers retain full custody of authentication and authorization audit trails.

6.6 MCP Server Security

Pathix exposes an in-product Model Context Protocol (MCP) server so that MCP-aware AI agents (Claude Desktop, Claude Code, Cursor, and similar clients) can query the parsed dependency and security graph. The MCP surface is governed by the same boundaries as the rest of the platform; this section states its security model precisely, including where authorization is intentionally coarse in 1.0.

Client authentication. MCP clients authenticate with a Microsoft Entra ID bearer token (JWT), validated by the **same `Microsoft.Identity.Web` JwtBearer middleware that protects the REST API** — there is no parallel authentication stack. The token is obtained via **OAuth 2.1 with PKCE (S256 mandatory)**. To let any HTTP-MCP client register and obtain a token by pasting a URL, Pathix runs its own OAuth Authorization Server in front of Entra (a Dynamic Client Registration proxy implementing RFC 8414 / 7591 / 6749 / 7636). **Critically, Pathix never issues its own JWTs and never sees user passwords:** the proxy mediates the OAuth flow and passes Entra's access token through *unchanged*, and `/mcp` validates that Entra token exactly as the REST layer does. Pathix holds no refresh-token signing keys and mints no tokens of its own.

Authorization — stated plainly. A distinct Entra scope, `Pathix.MCP`, gates the `/mcp` route: the route requires an authenticated user **and** the `Pathix.MCP` scope in the token's `scp` claim. The distinct scope exists so a tenant can audit MCP usage independently of browser-frontend usage in Entra sign-in logs. **There is no**

authorization below that scope check in 1.0 — no per-tool, per-table, per-row, or per-environment scoping. Any token carrying `Pathix.MCP` can reach every read tool across the entire scanned graph. This is the same single-tier posture as §6.4, carried through to the MCP: the customer's Entra controls govern *who can authenticate at all* and *who is granted the* `Pathix.MCP` *scope*; there is no second, in-app gate. In-app role-based authorization and MCP authorization scoping are the documented post-1.0 layer. For a forensics tool run by the customer's own security and administration team, this is a conscious, documented posture — but it is a real characteristic a reviewer should weigh, not one the document minimizes.

What contains the exposure. Three structural boundaries bound what a `Pathix.MCP`-scoped caller can do:

- **Metadata-only, end-to-end.** The MCP exposes the parsed dependency and security graph only — never Dataverse business record data. The §4 boundary holds through the MCP exactly as it holds through the REST API.
- **Actively defended OAuth flow.** Because Entra's own consent is suppressed (the upstream MCP client app registration is pre-authorized for `Pathix.MCP`), Pathix renders **its own consent screen** before any redirect leaves Pathix, showing the requesting client's `client_name` and `redirect_uri` HTML-encoded; the consent `confirm` POST is bound to the browser that rendered the page (HttpOnly, `SameSite=Strict` cookie, fixed-time compared) to defeat a CSRF auto-submit. An attacker cannot silently mint a `Pathix.MCP` token to their own `redirect_uri`. The OAuth proxy carries a **six-vector threat model with tests** — open redirect, PKCE downgrade, state binding, code single-use, cross-client injection, and consent-less DCR phishing.
- **AI narration is labeled untrusted.** Any AI-generated narration served through the MCP is labeled AI-generated and is to be treated by consuming agents as untrusted data — never as instructions or deterministic fact (§13.6).

Network, transport, and binding. The MCP is served over **HTTP / Streamable HTTP on the `/mcp` route of the existing `pathix-api` Azure Container App** — there is no separate container, image, network surface, or ingress. It therefore inherits whatever ingress controls the customer applies to the application (VNet restriction, private endpoint, Front Door + WAF — §8.1) with no application change. The **stdio transport is explicitly foreclosed** (Pathix is a hosted service in the customer's Azure, not a local subprocess; an `mcp-remote` bridge a customer runs locally speaks stdio only on the *client* side, while the Pathix server surface is HTTP-only). A **shared multi-tenant MCP is also foreclosed** — each customer runs its own MCP inside its own subscription against its own Entra. There is no vendor-managed multi-tenant MCP.

Gating. The `/mcp` route is mapped on every deployment — there is no "MCP on/off" feature toggle in 1.0 — but it is never openly reachable. It is gated three ways: (1) **auth-gated** — in any customer deployment it requires a valid `Pathix.MCP`-scoped Entra token; (2) **license-gated** — when the license is degraded or lapsed, the license gate short-circuits `/mcp` with a **403 *before* the transport is reached** (this is the mechanism behind the §2.2 / §14.2 "a lapsed license disables the MCP server" behavior); and (3) **customer setup required** — the customer must have provisioned the MCP client app registration at deploy time and wired an MCP client. Nothing connects on its own. In short: present on every deployment, but reachable only by an authenticated, `Pathix.MCP`-scoped caller, only while the license is valid, and only after the customer wires a client.

The MCP is therefore best understood as a capability with a documented hardening arc: the authentication, transport, and OAuth-phishing posture are mature today; sub-scope authorization within the MCP is the named next layer, not a control the product claims to have shipped.

7. Cryptography and Key Management

7.1 Encryption in Transit

All network communication between Pathix components and external services uses TLS 1.2 or higher, with cipher suites restricted to Microsoft Azure's default policy. This applies to Dataverse, Entra ID, Azure SQL Database, Azure Key Vault, and any customer-configured AI provider.

7.2 Encryption at Rest

All persistent data is encrypted at rest using Microsoft Azure platform-managed encryption:

- **Azure SQL Database** — Transparent Data Encryption (TDE) with Microsoft-managed keys (Customer-Managed Key option supported via Azure SQL configuration).
- **Azure Key Vault** — FIPS 140-2 Level 2 validated Hardware Security Modules.
- **Azure Container App** configuration and secrets — encrypted via Azure platform key management.
- **Azure Application Insights** — encrypted via Azure Monitor platform encryption.

7.3 Key Management

Pathix does not generate, manage, or escrow customer cryptographic keys directly. All cryptographic operations are delegated to Azure platform services. Customer-Managed Key (CMK) encryption is supported for Azure SQL Database and Azure Key Vault by configuring the underlying resources; Pathix imposes no constraints.

7.4 Vendor Signing Key

Pathix license JWTs are signed by an RSA private key held by the vendor (Pathix L.L.C.). The private key is held in a vendor password-manager vault with an Azure Key Vault secondary; the public key is embedded in the released container image (with an environment override available for key rotation). Customers do not handle, generate, or store this key.

Blast radius of a signing-key compromise: license forgery only. This key signs license JWTs (RS256) and nothing else — it holds no tenant access, no customer-data access, and no Dataverse credential. An attacker in possession of it could mint a valid entitlement or alter a license band or expiry, and that is the entire extent of the exposure: the key cannot read or write any customer environment. User authentication is Microsoft Entra (a separate stack) and Dataverse access is the Managed Identity holding the Pathix Scanner role (a separate stack); neither is reachable from the signing key.

7.5 Secret Storage on the Customer Side

The Pathix license token and any optional integration secrets are stored in **Azure Key Vault** in the customer's subscription. Application access to Key Vault is gated by Managed Identity and Azure RBAC. Secrets are never written to application configuration files, environment variables exposed to non-application processes, or log output.

8. Network Security

8.1 Network Topology

Pathix is deployed as an Azure Container App, which by default exposes a single HTTPS endpoint protected by Azure-managed TLS. Customers may optionally restrict ingress to specific virtual networks, enable Azure Front Door or Azure Application Gateway with Web Application Firewall (WAF), or restrict the application to private-endpoint access. These options are configurable at the Azure infrastructure layer and require no application changes.

8.2 Outbound Communication

During normal operation, Pathix initiates outbound HTTPS connections only to the following destinations:

Destination	Purpose	Conditionality
`login.microsoftonline.com`	Token acquisition for Entra ID and Managed Identity	Always
`{tenant}.crm.dynamics.com`	Dataverse metadata reads	Always
`{customer}.vault.azure.net`	Secret retrieval	Always
`{customer}.database.windows.net`	Application database	Always
`api.powerapps.com`	Canvas application metadata (Power Apps Admin API)	When canvas-app scanning enabled
Customer-configured AI endpoint	Optional AI features (BYOK)	Customer-enabled
Customer-configured telemetry / crash-report endpoint (e.g., Application Insights, customer-owned)	Operational telemetry	Customer-enabled / opt-in

All outbound endpoints are loggable and may be restricted by the customer's egress policy. **Pathix L.L.C. operates no outbound destination required for normal product function.** License validation is air-gap-by-default (Section 14.2); a vendor-operated endpoint is not required for the product to operate.

9. Application Security

9.1 Secure Coding Standards

Pathix is developed in C# (.NET 10) and TypeScript. The codebase enforces:

- Nullable reference types across the entire backend codebase, eliminating broad classes of null-dereference defects.
- Treat-warnings-as-errors compilation, including security-relevant analyzers.
- TypeScript strict mode in the frontend (strict null checks, no implicit any).
- Parameterized queries and Object-Relational Mapper (ORM) usage in all database access; no string concatenation of SQL.
- Output encoding via React's default JSX escaping; raw HTML insertion (`dangerouslySetInnerHTML`) is forbidden by lint rule.

9.2 Common Vulnerability Mitigations

Vulnerability class	Mitigation
SQL injection	Parameterized queries via Entity Framework Core and Dapper; no dynamic SQL construction
Cross-site scripting (XSS)	React JSX escaping by default; Content Security Policy headers
Cross-site request forgery (CSRF)	SameSite cookies; bearer-token authentication on API endpoints
Insecure deserialization	`System.Text.Json` with allow-list typing; no `BinaryFormatter` usage
Server-side request forgery (SSRF)	Outbound HTTP restricted to a compile-time allow-list of endpoints
Authentication bypass	All API endpoints require valid Entra ID bearer tokens, validated against Microsoft public keys
Sensitive data exposure	Metadata-only boundary (§4); structured-logging filters reject record-data fields

9.3 Input Validation

All API request bodies are validated against typed schemas. Frontend forms use schema-based validation. Server-side validation is authoritative; client-side validation is for usability only.

10. Logging, Monitoring, and Auditability

10.1 Structured Logging

All application events are emitted as structured log records (Serilog) and routed to the customer's Azure Application Insights instance. Log records include correlation identifiers, the authenticated principal (when applicable), the operation performed, and the outcome. The log emission infrastructure enforces:

- Customer record data is never logged (enforced at the SDK wrapper layer — there is no source of record data in the process).
- Authentication failures, authorization failures, and configuration changes are logged at Warning or higher severity.
- Stack traces are sanitized to remove parameter values from exception serialization.

10.2 Audit Trail

Audit-trail events emitted by the application include: user authentication (success and failure), scan initiation/completion/failure, configuration changes (environment registration, schedule changes, AI provider configuration), license validation events, and Dataverse SDK calls (operation, target, result — never response payloads).

10.3 Log Custody and Retention

Because all logs are emitted to a customer-owned Application Insights instance, log custody, retention, and access control rest entirely with the customer. Pathix does not exfiltrate operational logs to vendor infrastructure.

11. Vulnerability Management and Secure Development

11.1 Software Development Lifecycle

Pathix follows a documented secure software development lifecycle:

- Architecture Decision Records (ADRs) for all security-relevant design decisions, with explicit amendment trails when posture changes.
- Mandatory peer review for all code changes via pull request.
- Continuous integration with unit and integration tests.
- Static application security testing via.NET analyzers and ESLint security plugins.
- Dependency vulnerability scanning via GitHub Dependabot with automated pull requests for vulnerable transitive dependencies.

11.2 Vulnerability Disclosure

Suspected vulnerabilities may be reported to `security@pathix.app`. Reports are acknowledged within two business days and triaged within five business days. Critical-severity findings affecting deployed customers receive remediation guidance and patched releases on an expedited schedule (target: within fifteen calendar days of validated discovery). A `.well-known/security.txt` declaration (RFC 9116) on `pathix.app` provides the machine-readable pointer to this policy.

11.3 Patch Cadence

Because Pathix is customer-deployed, patch cadence is jointly governed by the vendor (release cadence) and the customer (deployment cadence). Vendor releases are versioned semantically. Critical security patches are flagged as such in release notes. Customers retain full control over when patched images are deployed.

11.4 Penetration Testing

Pathix releases are subjected to structured security review prior to general availability. Customers may, at their discretion, conduct independent penetration testing of their deployed Pathix instance subject to Microsoft Azure's customer penetration-testing policy. The vendor will collaborate with customers conducting such testing under reasonable disclosure terms.

12. Third-Party Components and Supply Chain

12.1 Dependency Posture

Pathix's dependency posture is deliberately conservative. New runtime third-party dependencies require explicit approval before introduction. The full set of runtime dependencies is enumerated in source-controlled package manifests (NuGet for.NET, npm for the frontend) and is subject to automated vulnerability scanning.

12.2 Notable Runtime Dependencies

Dependency	Source	Function
`Microsoft.PowerPlatform.Dataverse.Client`	Microsoft	Dataverse SDK
`Microsoft.EntityFrameworkCore`	Microsoft	Object-relational mapping
`Microsoft.Identity.Client` / `Azure.Identity`	Microsoft	Authentication
`Microsoft.PowerFx.Core`	Microsoft	PowerFx parsing
`Polly`	App vNext (OSS)	Resilience and retry policies
`Serilog`	OSS	Structured logging
`Dapper`	Stack Exchange (OSS)	Lightweight data access
`ICSharpCode.Decompiler`	OSS (MIT)	Plugin assembly decompilation for the optional AI plugin-summarization feature (§13.3)
`Mono.Cecil`	OSS	Plugin IL static analysis (deterministic walker)
`Next.js` / `React`	Vercel / Meta (OSS)	Frontend framework
`Tailwind CSS`	OSS	User interface styling

12.3 Software Bill of Materials

A complete Software Bill of Materials (SBOM) in **CycloneDX** format ships with Pathix 1.0: SBOM generation runs in the release pipeline, and the SBOM is published per release on a forward basis. Customers may also request the current SBOM via `security@pathix.app`.

13. Artificial Intelligence Use

13.1 Bring Your Own Key, Off by Default

AI features within Pathix are **optional** and operate exclusively on a **Bring Your Own Key (BYOK)** model. The customer configures one of:

- Microsoft Azure OpenAI Service
- Anthropic API
- OpenAI API

All AI requests originate from the customer's Container App and terminate at the AI provider configured by the customer, **under the customer's agreement with that provider.** The vendor does not proxy, broker, intercept, store, or have visibility into AI provider credentials, AI requests, or AI responses.

13.2 Graceful Degradation

The Pathix deterministic core operates at full fidelity with zero AI. AI is an additive layer: it narrates the graph, supports natural-language queries, and contributes evidence-coupled candidate edges (Section 13.4). It is never a precondition for analytical capability.

13.3 Plugin-Summarization Sweep — Opt-In, Per-Publisher Scope

One AI feature is an optional **post-scan plugin-summarization sweep** that decompiles plugin assemblies registered in the customer's Dataverse environment — including **third-party / ISV-shipped assemblies** — and feeds the decompiled C# to the customer's BYOK AI provider to produce a behavioral summary. The decompiled bytes are discarded; only the natural-language summary is persisted (in the customer's own database, never received by the vendor).

This feature is **off by default**. Enabling it is a per-environment customer action that requires:

- **Affirmative, recorded consent** confirming, for the publishers the customer has not excluded, that decompilation is permitted under their vendor agreements.
- **Per-publisher, customer-driven exclusion.** Pathix surfaces every publisher detected in the environment; the customer marks third-party publishers to exclude. Pathix does **not** infer authorship from packaging (a "managed" solution is just as often the customer's own code as a third-party product), so exclusion is always a customer judgment.

The decompilation executes inside the customer's tenant with the customer's AI key. Pathix L.L.C. never receives the assemblies or their decompiled form.

13.4 AI Output Boundary

AI output **never overrides the deterministic structural graph.** The deterministic walker (Mono.Cecil-based IL static analysis, plus the parser surfaces for workflows, flows, scripts, formulas, etc.) is the source of truth for dependency edges labeled `High`, `Medium`, or `Inferred`.

Where AI surfaces a relationship the deterministic walker did not — a write the walker missed entirely, or an `Inferred`-tier write whose target the walker could not resolve — the AI's contribution is recorded as an `AiDerived` **edge**, evidence-coupled (the narration / decompiled-IL snippet that justifies it is attached), distinctly labeled, and **net-new only** (never produced where a deterministic edge already exists for the same component/target tuple). `AiDerived` edges sort below `Medium` in display, are surfaced as labeled candidates with attached evidence, and are first-class filterable in exports and MCP responses so downstream consumers can distinguish AI-proposed from deterministic relationships. This is the structural enforcement of the principle: **AI proposes; the walker proves; a named human confirms.** The labeling is in place today, and a human-validation overlay lets a reviewer accept, dismiss, or restore each `AiDerived` edge, with that decision tracked independently of the model's confidence.

13.5 Customer Data in AI Prompts

AI prompts are constructed to send structural metadata (component names, action types, target columns) rather than raw definition payloads where possible. Where raw payloads are required to answer a question — most notably the decompiled C# for the plugin-summarization sweep — the BYOK model ensures such data flows only to the customer's chosen provider, within the customer's existing AI compliance posture.

13.6 Prompt Injection and Adversarial AI Input

The AI layer described above is deliberately pointed at the **least-trustworthy content in an environment** — decompiled plugin logic, form-script JavaScript, HTML web resources, classic-workflow definitions, and modern-flow JSON. Every string literal, comment, and identifier in that input is controllable by whoever authored the component, including third-party ISVs. This is the textbook condition for **indirect (stored) prompt injection**: adversarial text that attempts either to steer Pathix's own summarization (for example, a malicious plugin "arguing" that it performs no writes) or to plant a payload in the persisted narration that later steers a customer's own downstream AI agent.

Pathix treats this as a first-class threat and takes an explicit, honest position: **prompt injection is contained, not solved.** No system can guarantee a language model is never misled by its input. What Pathix guarantees today is narrower and structural: **an injection cannot move or forge a deterministic dependency fact, and cannot exfiltrate customer record data.** A forensics tool must also **never sanitize** a suspicious string out of its structured output — the unusual string is itself evidence — so the defense is delimiting, labeling, and cross-checking, never scrubbing.

Controls in place today (defense in depth):

- **No sensitive data in the prompt.** Under the metadata-only boundary (Section 13.5), the model never receives customer record values, so there is no business data in the AI path to disclose. The worst a successful injection can do is bias narration text or attempt to steer a downstream agent; it cannot leak records.
- **Non-forgeable untrusted-content framing.** Every untrusted block is wrapped between per-request, cryptographically-random delimiters and presented to the model as third-party data to analyze, never as instructions to follow. Because the random token cannot be guessed from the scanned content, an injected literal cannot forge a closing delimiter to "break out" of the data region. This framing is applied across all five AI-analyzed surfaces.
- **Output firewall.** The model's response must parse as the canonical structured schema; non-schema output, and any prose the model wraps around it, is dropped. A structural-escape attempt yields a failed summary, not a poisoned one.
- **Deterministic-edge armor.** As described in Section 13.4, AI-derived edges are a distinct, hard-pinned, evidence-coupled, net-new-only tier. The model cannot promote its claim to a deterministic tier, cannot invent an edge to a non-existent column (non-resolving edges are dropped), and cannot override a deterministically-proven edge.
- **Bounded input.** Untrusted content is size-capped before the AI call, with a visible truncation marker rather than silent loss, bounding both context-window failures and metered AI cost.

- **Opt-in, BYOK, default-off.** The AI layer is off by default and uses the customer's own model and key. There is no shared, multi-tenant model surface; the blast radius of any injection is the customer's own tenant and their own model.

Residual risk, stated plainly:

- **Biased narration.** Narration is model-generated free text. A payload that produces structurally valid output can still bias the wording served to humans and downstream agents. Mitigation relies on the model's instruction/data separation holding, which is probabilistic, not a proof. This is inherent to AI summarization of untrusted input.
- **Silent under-report.** A deterministic **under-report cross-check** — which would raise an integrity flag whenever narration omits a dependency edge the deterministic walker has already proven — is **designed but not yet shipped.** Until it ships, an injection that talks the narrator into omitting a known write, while still producing valid output, is not automatically caught. The deterministic graph is computed independently and is unaffected, so the proven edges remain available; the risk is that a reader who trusts prose over the graph could be misled. This cross-check is a gate on any future move to enable the AI feature by default.
- **Downstream consumers.** Narration served through the Pathix MCP interface is always labeled AI-generated and should be treated by consuming agents as untrusted data, never as instructions or deterministic fact.

External mapping. This posture maps to the **OWASP Top 10 for LLM Applications (2025)**: LLM01 (Prompt Injection), addressed by the framing controls above, with the under-report cross-check as planned containment; LLM05 (Improper Output Handling), by the schema firewall and hard-pinned confidence; LLM02 (Sensitive Information Disclosure), by the metadata-only boundary; LLM06 (Excessive Agency), by the deterministic-edge armor; LLM09 (Misinformation), by the AI-derived labeling; and LLM10 (Unbounded Consumption), by the input cap.

14. Telemetry, Licensing, and Privacy

Pathix deliberately separates three categories that are often conflated under the word "telemetry."

14.1 Categories at a Glance

Category	1.0 posture	Content
License validation	Mandatory — air-gapped by default	License token / entitlement, tenant identifier, version, timestamp
Anonymous usage telemetry	**Not collected.** No implicit collection, no default-on toggle.	Standing privacy commitment; see §14.3.
Crash diagnostics	**Customer-initiated, manual.** No automatic transmission in 1.0.	A local log export the customer generates and emails to support.

14.2 License Validation — Air-Gapped by Default

License validation is performed by **local cryptographic signature check** against the embedded vendor public key. No network call is required. Online phone-home is optional plumbing — used only for renewal hints and soft revocation — and is not the path on which validation depends. Customers with no outbound internet (some government, defense, and financial deployments) deploy and run Pathix with no network changes to procurement.

The license payload is intentionally small: license token / entitlement record, tenant identifier, version, timestamp. **No usage data, no customer data, no findings data** crosses this surface.

A lapsed or invalid license **soft-degrades, never hard-bricks.** The degraded behavior blocks new scans and disables the MCP server but **never deletes, locks, or gates Customer's access to data already scanned.** The data is in the customer's own SQL database and remains the customer's. This is the answer to the common reviewer question: *what happens to our data if we stop paying?*

14.3 Anonymous Usage Telemetry — Not Collected

Pathix does not collect anonymous usage telemetry. There is no implicit collection, no default-on toggle, and no background channel to the vendor.

As a standing privacy commitment, the following are never collected, even hashed:

- Table names, column names, role names, plugin names, workflow names, or any identifier originating from customer customization.
- Customer or tenant identifiers beyond the license identifier.
- Customer record data of any kind.
- Finding contents of any kind.
- Any string originating from customer customization or environment-controlled content.

14.4 Crash and Support Diagnostics (1.0 — Customer-Initiated)

Pathix 1.0 does **not** transmit crash diagnostics automatically. There is no auto-uploader, no opt-in flag, no background diagnostic channel to the vendor.

When a problem warrants support engagement, the customer generates a **local log export** (a zip of the relevant logs and scan-phase events from their own Pathix instance) and emails it to the support contact at `help@pathix.app`. The customer chooses what to send, when to send it, and reviews the content before sending.

By design, Pathix logs component identifiers and error *types*, never raw content from a failed parse, so this export respects the metadata-only boundary by construction. The vendor receives nothing it would not have received under an opt-in sanitized crash-report flow — only the customer is in the loop earlier.

15. Business Continuity and Resilience

15.1 Application Resilience

All outbound calls to Microsoft Dataverse are protected by a Polly v8 resilience pipeline implementing exponential backoff with jitter, retry on transient errors (HTTP 429, 502, 503, 504), and per-attempt timeouts. Authentication failures, authorization failures, and schema errors are not retried. The Dataverse SDK wrapper centralizes retry classification.

15.2 Backup and Recovery

Backup and recovery of the Pathix database are governed by Azure SQL Database's automated backup capability, configured by the customer. Pathix does not require additional backup tooling. Because each Pathix scan produces a complete, immutable snapshot of the scanned environment, point-in-time recovery to a previous scan is also supported as a native data-model property.

15.3 Disaster Recovery

Disaster-recovery posture is determined by the Azure region(s) selected by the customer at deployment time. Multi-region deployment is supported via Azure-native geo-replication. The vendor does not impose RTO/RPO commitments because the underlying infrastructure is owned and operated by the customer.

16. Compliance Posture

16.1 What Determines the Posture

Pathix's compliance posture is determined principally by two architectural commitments — the metadata-only data boundary (§4) and customer-hosted deployment (§3). These two commitments narrow the scope of regulatory frameworks that attach to the Pathix platform itself, while preserving the customer's ability to satisfy regulatory requirements in their broader environment.

16.2 Framework Mapping

Framework	Pathix posture
SOC 2 Type II	Pathix L.L.C. has no production hosting infrastructure that processes customer data. The customer's Azure subscription provides the SOC 2 control environment. Pathix-generated findings map to CC6 (Logical Access), CC7 (System Operations), and CC8 (Change Management) controls.
ISO/IEC 27001:2022	The customer's Azure subscription provides ISMS scope. Pathix supports A.5 (Organizational), A.8 (Asset Management), and A.9 (Access Control) controls through visibility into the Dataverse environment.
NIST SP 800-53	Findings mappable to AC (Access Control), AU (Audit and Accountability), CM (Configuration Management), and SI (System and Information Integrity) control families.
GDPR (Regulation 2016/679)	Pathix holds **no business record content**, so GDPR data-subject rights *against record content* do not attach to Pathix. Pathix does store administrative principal identifiers (display names, UPNs) — personal data — in the customer's own tenant; **the customer remains the controller.** Pathix L.L.C. receives no personal data and is, in its documented position, a software supplier rather than a data processor.
HIPAA (45 CFR §§ 160, 164)	A Business Associate Agreement is not required because **the vendor does not receive customer data** — Pathix executes within the customer's Azure tenant. PHI handling, if any, remains entirely within the customer's environment and subject to the customer's own HIPAA program.
PCI-DSS v4.0	Pathix does not process cardholder data. Out of cardholder-data-environment scope by architecture.

16.3 What This Section Does Not Claim

Notably absent from the table is the phrase "*Pathix does not process personal data.*" It is not the right claim. The right claim — used above and in the privacy commitment — is that the **vendor receives no personal data**, while the **product holds limited personal data about administrative principals in the customer's own tenant.** A careful reviewer will probe an over-claim here; Pathix takes the precise framing on purpose.

16.4 Sub-Processors

Pathix L.L.C. engages zero vendor-operated sub-processors that touch customer data. Because the product executes within the customer's own Azure tenant and the vendor receives no customer data (§3, §5), there is no vendor-side processing to sub-contract: there is no vendor-operated error-reporting service, content-delivery network, analytics SDK, crash reporter, email service, or message queue anywhere in the product's runtime data path.

The only optional recipients of data are **the customer's own services**, engaged at the customer's election:

Optional recipient	When	Whose resource	What it receives
BYOK AI provider (Azure OpenAI / Anthropic / OpenAI)	Only if AI features are enabled	The customer's own account and key	Structural metadata and, for the plugin-summarization sweep, decompiled plugin C# (§13.3, §13.5)
Application Insights	Only if the customer adds the sink	The customer's own subscription	Operational logs and metrics (§10)

In both cases the **customer is the controller and the contracting party**; Pathix L.L.C. is not a party to that data flow and never receives the data. They are therefore **customer data recipients, not Pathix sub-processors** in the GDPR-processor sense.

The following are **not sub-processors** and are itemized here to forestall the question: **GitHub Container Registry** (image distribution — container images are *pulled* by the customer at deploy time; it is not a runtime data processor); **license validation** (an air-gapped local signature check with no vendor endpoint — §14.2); and the always-on Microsoft platform destinations in §8.2 (`login.microsoftonline.com` , `{tenant}.crm.dynamics.com` , `{customer}.vault.azure.net` , `{customer}.database.windows.net` , and `api.powerapps.com` when canvas scanning is enabled), which are **Microsoft platform services operating inside the customer's own tenant**, not third-party sub-processors.

17. Customer Responsibilities (Shared Responsibility Model)

Because Pathix is customer-hosted, the responsibility model differs from a typical SaaS engagement.

Domain	Vendor	Customer
Application code	✓	—
Security patches and releases	✓ (publish)	✓ (deploy)
Azure subscription and billing	—	✓
Network configuration	—	✓
Database backup and DR	—	✓ (Azure-native)
Identity and access management	—	✓ (Entra ID, customer-controlled)
Dataverse environment hygiene	—	✓
AI provider relationship and billing	—	✓ (BYOK)
License compliance	✓ (issue)	✓ (operate within terms)
End-user training	✓ (documentation)	✓ (delivery)
Incident response within customer tenancy	—	✓
Vulnerability disclosure to vendor	—	✓

18. Incident Response

18.1 Customer-Side Incident Response

Because Pathix executes within the customer's Azure subscription and processes only metadata that the customer's Dataverse environment already exposes to authenticated administrators, security incidents involving the Pathix platform are, in the first instance, **customer incidents** to be handled in accordance with the customer's established incident-response procedures.

18.2 Vendor Coordination

In the event a customer identifies a security incident attributable to Pathix software (as distinct from infrastructure or configuration), the customer is encouraged to engage the vendor via the disclosure channel in §11.2. The vendor will coordinate with the customer to:

- Validate and characterize the incident.
- Develop and ship remediation in software where required.
- Notify other customers of relevant indicators of compromise or required actions, where the incident has cross-customer implications.

18.3 Breach Notification

Because the vendor does not host customer data, statutory breach-notification obligations triggered by unauthorized access to customer data rest with the customer as the controller. The vendor's breach-notification obligations are limited to circumstances in which vendor-controlled systems (e.g., the vendor source repositories or the signing-key infrastructure) are compromised in a manner relevant to customer security.

19. Conclusion

Pathix's security architecture is built on a small number of deliberately constraining design commitments: customer-hosted deployment, metadata-only data access, passwordless authentication, and bring-your-own-key AI integration that is off by default. These commitments collectively reduce the security and compliance surface that the Pathix platform introduces into a customer's environment, while preserving full analytical capability for the operational and security challenges Pathix is designed to address.

This whitepaper is deliberately precise about what Pathix holds and what it does not — including the personal data Pathix *does* store about administrative principals in the customer's own tenant. The vendor invites independent review and will support customer security assessments with documentation, source-of-truth references, and direct engineering engagement as required.

Contact. For security inquiries, vulnerability reports, or whitepaper clarifications: `security@pathix.app`. For procurement, compliance, and enterprise security questionnaires: `info@pathix.app`.

Appendix A: Pathix Scanner role — exact privilege matrix

This appendix reproduces the Pathix Scanner role at the level of detail a security reviewer needs to diff it against their own environment. The **authoritative privilege set ships in the Pathix Scanner managed solution** (`PathixScanner_<version>_managed.zip`) and is auditable in the Power Platform Admin Center prior to assignment (the customer-side verification path in §4.5, step 1). The matrix below is the canonical statement of what that solution grants; a customer confirms it against the imported role in their own PPAC, which is the real audit surface.

Every privilege below is a Read privilege granted at Organization (Global) scope. Pathix requests no write, create, delete, append, append-to, assign, or share privilege anywhere. Privilege names are **case-sensitive and reproduced exactly** as Dataverse names them — including the lowercase `prvReadappaction`, the casing of `prvReadCustomAPI`, and three platform naming quirks called out inline (`prvReadQuery` gates `savedquery`; `prvReadUser` gates `systemuser`; `prvReadSavedQueryVisualizations` is plural).

A.1 The 29 requested Read privileges

#	Privilege name (exact)	Entity / surface gated	Scope	Operation
1	`prvReadPluginAssembly`	pluginassembly	Organization	Read
2	`prvReadPluginType`	plugintype	Organization	Read
3	`prvReadSdkMessage`	sdkmessage	Organization	Read
4	`prvReadSdkMessageProcessingStep`	sdkmessageprocessingstep	Organization	Read
5	`prvReadSdkMessageProcessingStepImage`	sdkmessageprocessingstepimage	Organization	Read
6	`prvReadWorkflow`	workflow	Organization	Read
7	`prvReadWebResource`	webresource	Organization	Read
8	`prvReadSystemForm`	systemform	Organization	Read
9	`prvReadSavedQueryVisualizations`	savedqueryvisualization (privilege name is **plural**)	Organization	Read
10	`prvReadCustomization`	customization umbrella — covers sitemap, ribboncustomization, customcontrol	Organization	Read
11	`prvReadQuery`	savedquery (named `prvReadQuery`, **not** `prvReadSavedQuery`)	Organization	Read
12	`prvReadappaction`	appaction (privilege name is **lowercase**)	Organization	Read
13	`prvReadmsdyn_dataflow`	msdyn_dataflow	Organization	Read
14	`prvReadconnectionreference`	connectionreference	Organization	Read
15	`prvReadServiceEndpoint`	serviceendpoint — covers service endpoints **and** webhooks (discriminated by `contracttype`)	Organization	Read
16	`prvReadCanvasApp`	canvasapp	Organization	Read
17	`prvReadPublisher`	publisher	Organization	Read
18	`prvReadCustomAPI`	customapi	Organization	Read
19	`prvReadRole`	role	Organization	Read
20	`prvReadFieldSecurityProfile`	fieldsecurityprofile (gates fieldpermission reads; there is no	Organization	Read

#	Privilege name (exact)	Entity / surface gated	Scope	Operation
		standalone `prvReadFieldPermission`)		
21	`prvReadTeam`	team	Organization	Read
22	`prvReadOrganization`	organization	Organization	Read
23	`prvReadPosition`	position	Organization	Read
24	`prvReadSolution`	solution	Organization	Read
25	`prvReadUser`	systemuser (named `prvReadUser`, **not** `prvReadSystemUser`)	Organization	Read
26	`prvReadAnyPrivilegeEntity`	umbrella — configuration-entity reads that have no standalone privilege (not a single table)	Organization	Read
27	`prvReadEntity`	entity metadata	Organization	Read
28	`prvReadAttribute`	attribute metadata	Organization	Read
29	`prvReadRelationship`	relationship metadata	Organization	Read

A.2 The metadata layer (`RetrieveAllEntitiesRequest`)

Rows 27–29 — `prvReadEntity`, `prvReadAttribute`, `prvReadRelationship` — are the **metadata-layer Read privileges**. They authorize the Dataverse metadata-service path (`RetrieveAllEntitiesRequest`, `RetrieveEntityRequest`, and related metadata messages). This is a **privilege grant, not a per-message access-control entry**: Dataverse gates metadata-service messages through these three privileges, and there is no separate per-message privilege for the metadata service. This is distinct from the plugin-family rows 3–5 (`prvReadSdkMessage*`), which authorize reading the `sdkmessage` and processing-step **records** (i.e., plugin step registrations) — a different surface from the metadata catalog.

A.3 Platform-injected privileges (not requested by Pathix)

Where SharePoint document-management integration is enabled in an environment, **Microsoft automatically appends the following SharePoint privileges to *all* custom security roles** — the Pathix Scanner role included — and re-adds them if they are removed. They are **platform-added, never present in Pathix's request, and never exercised by Pathix.** This is the same caveat described in §6.3 and the §4.5 verification note; the privileges are expected, not a finding.

Privilege name	Surface	Scope	Note
`prvReadSharePointData`	SharePoint data	Global	Platform-injected by SharePoint document-management; not requested by Pathix
`prvReadSharePointDocument`	SharePoint document	Global	Platform-injected; not requested by Pathix
`prvCreateSharePointData`	SharePoint data	Global	Platform-injected (a Create privilege — note it is **not** a Read); not requested by Pathix
`prvWriteSharePointData`	SharePoint data	Global	Platform-injected (a Write privilege — note it is **not** a Read); not requested by Pathix

Pathix reads metadata through the SDK wrapper and never accesses SharePoint document content. A reviewer diffing the imported role against this appendix will see these four privileges present and should attribute them to the platform, not to Pathix.

A.4 Considered and excluded

The following privileges were evaluated and are **deliberately not granted**; each is listed with the reason, so a reviewer sees they were a conscious omission rather than an oversight:

Privilege	Why excluded
`prvReadPrincipalObjectAccess`	Principal Object Access (POA) is gated by a share privilege (`prvShare<Entity>`) on the parent table, not by a Read privilege — and Pathix requests no share privilege. POA is therefore not part of the role.
`prvReadSdkMessageFilter`	Not present in the privilege catalog; `sdkmessagefilter` access is implicit via `prvReadSdkMessage` .
`prvReadRibbonCustomization`	Not a standalone catalog privilege; covered by the `prvReadCustomization` umbrella (row 10).
`prvReadCustomControl`	Not a standalone catalog privilege; covered by the `prvReadCustomization` umbrella (row 10).

Appendix B: Glossary of Terms

Term	Definition
ARM / Bicep	Azure Resource Manager / Bicep — declarative infrastructure-as-code formats used to deploy Azure resources.
BYOK	Bring Your Own Key — a deployment model in which the customer supplies credentials for an integrated third-party service.
Dataverse	Microsoft Dataverse — the underlying data platform for Microsoft Dynamics 365 and Microsoft Power Platform.
Entra ID	Microsoft Entra ID — the cloud-based identity and access management service formerly known as Azure Active Directory.
Managed Identity	An Azure-managed service identity used for passwordless authentication between Azure services.
MCP	Model Context Protocol — an open protocol for AI agents to call external tools/data; Pathix exposes an in-product MCP server over its dependency/security graph.
Metadata	In the Pathix context, structural and configurational information about a Dataverse environment, distinct from record-level business data.
PCF	PowerApps Component Framework — Microsoft's framework for custom user-interface components within Power Apps.
PowerFx	Microsoft's low-code formula language used in Power Apps canvas applications and formula columns.
RBAC	Role-Based Access Control — an authorization model in which permissions are granted to roles, and roles are assigned to principals.
SAST	Static Application Security Testing — automated analysis of source code to identify security defects.
SBOM	Software Bill of Materials — a structured inventory of all software components in a release.
TDE	Transparent Data Encryption — Azure SQL Database's encryption-at-rest feature.
UPN	User Principal Name — the canonical Entra ID identifier for a user (e.g., `alice@contoso.com`).