Service Integration Framework

Provided to Aetna for I90 Integration Framework

Service Level Agreement (SLA) and Integration Protocol governing the partnership between the Service Provider and the Service Consumer. Comprehensive framework formalizing:

* **Governance & Personnel:** Defines key roles, responsibilities, and specific communication protocols for engineering, support, and management teams.
* **Service Level Guarantees:** Establishes binding targets for service availability, performance, and support responsiveness.
* **Integrated Program Management:** Outlines a formal process for synchronizing Project Management Offices (PMOs) on major joint initiatives.
* **Full Technical Lifecycle:** Details the complete operational process from initial onboarding and runtime communication to service evolution, versioning, and eventual retirement.

Service Integration Framework

|  |
| --- |
| **Document Version:** 6.0 (Final) |
| **Effective Date:** July 25, 2025 |
| **Last Updated:** Friday, July 25, 2025 |
| **Origination:** Aetna Digital |

# Section 1: Introduction & Core Concepts

* ***Description:*** *This initial section serves as the foundation for the entire agreement. It defines the key entities, services, and terms that will be used throughout the document.*
* ***Goal:*** *To establish a common vocabulary and eliminate ambiguity, ensuring both the Service Provider and Service Consumer understand their roles and the components being discussed in precisely the same way.*

**1.1. Preamble** This document constitutes an agreement between the Service Provider and the Service Consumer, governing the technical integration, operational support, and lifecycle management of the services provided herein. Adherence to this protocol is mandatory for maintaining a healthy, predictable, and reliable partnership.

**1.2. Core Definitions**

* **Service Provider:** The entity that owns, develops, and maintains the services.
* **Service Consumer:** The application or system that integrates with and utilizes the services.
* **Service Contract:** The complete set of agreements between the parties, including this document and any commercial terms.
* **REST API Service:** A data-centric service providing access to core business entities and operations via standard HTTP methods (GET, POST, PUT, DELETE) for programmatic interaction.
* **UX/Data Block Service:** A specialized, opinionated service that provides a structured, often pre-configured, block of data and metadata intended for direct rendering within a consumer's user experience.

# Section 2: Governance, Personnel, & Service Level Agreement (SLA)

* ***Description:*** *This section defines the human layer of the partnership and the quality guarantees of the service. It outlines key personnel roles, their responsibilities, the measurable standards for service availability and performance, and the specific communication protocols for different functional teams during ongoing operations.*
* ***Goal:*** *To create clear lines of responsibility, establish predictable communication pathways for all scenarios, provide the Consumer with a formal guarantee of service quality and support responsiveness, and ensure that the right people are talking to each other in the right way at the right time.*

### 2.1 Stakeholder Responsibility & Communication Matrix

| **Stakeholder Role (Provider / Consumer)** | **Key Responsibilities** | **RACI Mapping (See Key Below)** | **Primary Communication Channels** |
| --- | --- | --- | --- |
| **Executive Sponsor** | - Champions the partnership internally. - Secures strategic resources and budget. - Final point of escalation for high-level disputes. | **A:** Joint Project Charter Approval **I:** Contractual Agreement, SLA Performance, Deprecation Notices, QBR | Scheduled Executive Meetings, Email Escalations |
| **Account Manager / Primary Business Contact** | - Manages the overall business relationship. - Handles all commercial and contractual matters. - Organizes and leads Quarterly Business Reviews (QBRs). | **A:** Contractual Agreement, SLA Performance & Remedies, QBR **R:** Deprecation Notices **C:** Joint Project Charter Approval | Email, Scheduled Meetings (QBR, Monthly Sync) |
| **Program Lead** | - Owns the product/business vision for joint projects. - Defines project goals and success criteria. - Serves as a key decision-maker for project scope. | **A:** Integrated Master Schedule (IMS) **R:** Joint Project Charter Approval **C:** SLA Performance, Technical Integration **I:** S1 Incidents, Deprecation Notices | Email, Project Steering Committee Meetings |
| **Project Manager (PM)** | - Manages internal project plans, resources, and timelines. - Synchronizes with counterpart on the Integrated Master Schedule (IMS). - Manages the joint risk register and change control process. | **R:** IMS Management, Technical Onboarding **C:** Joint Project Charter Approval **I:** S1 Incidents | Shared Project Management Tools (Jira, Asana), Weekly Tactical Syncs, Email |
| **Technical Integration Lead / Primary Technical Contact** | - Main technical point of contact for the integration. - Provides architectural guidance and resolves deep technical issues. - Receives and disseminates all lifecycle communications. | **A:** Technical Integration & Onboarding **R:** S1 Incident Response **C:** IMS Management, QBR **I:** SLA Performance, Deprecation Notices | Shared Slack/Teams Channel, Ad-hoc Video Calls, Developer Forums, Email |
| **Support Engineering Team / On-Call Engineering Team** | - Manages and resolves end-user support tickets. - First line of defense for incident detection and triage. - Manages the technical response to all incidents. | **A:** S1 Incident Response **C:** SLA Performance Reports, Technical Onboarding **I:** Deprecation Notices | Support Portal/Ticket System, Phone (Hotline), On-Call Alerts, Email |

**RACI Key:**

* **R = Responsible:** The person/team who does the work.
* **A = Accountable:** The person ultimately answerable for the work (the owner).
* **C = Consulted:** Provides input and expertise (two-way communication).
* **I = Informed:** Kept up-to-date on progress (one-way communication).

**2.2. Service & Personnel Availability**

**2.2.1. Service Level Agreement (SLA)**

The Provider shall commit to the following service levels for all **Production** environments. These levels pertain to the automated services and the formal support response channels. Remedies for failing to meet these levels are defined in the commercial agreement.

* **API Availability & Uptime:**
  + **Target:** 99.9% monthly uptime for all service endpoints.
  + **Exclusions:** Scheduled Maintenance, outages caused by the Consumer, or failures of third parties beyond the Provider's reasonable control.
* **API Performance & Latency:**
  + **Target:** 95% of all internal API processing times will be completed in under 500ms. Latency is measured at the Provider's network edge.
* **Support Response Times:**

| **Severity** | **Description** | **Target Response Time** | **Channels** |
| --- | --- | --- | --- |
| **Severity 1: Critical** | Complete service outage or critical functionality failure. | **1 Hour** (24/7/365) | Phone call to On-Call Contact, Email |
| **Severity 2: High** | Significant performance degradation or major feature failure. | **4 Business Hours** | Support Portal, Email |
| **Severity 3: Medium** | Minor feature failure or issue affecting a limited subset of users. | **1 Business Day** | Support Portal, Email |
| **Severity 4: Low** | General questions or documentation requests. | **2 Business Days** | Support Portal, Email |

s

**2.2.2. Human Resource Availability**

This section outlines the agreed-upon expectations for the availability of key personnel from both the Provider and Consumer for collaborative activities that fall outside the scope of the ticket-based support SLA defined above.

Both parties agree to make the following personnel available to their counterparts on an **as-needed basis** to support formally defined Joint Projects, resolve complex escalations, and facilitate the ongoing partnership. Availability is generally defined as being accessible for scheduled meetings and ad-hoc discussions during standard business hours (9:00 AM to 5:00 PM) in their respective time zones, Monday through Friday, excluding national holidays.

* **Management (Account Managers, Program Leads, Project Managers):** Available for scheduled governance meetings (e.g., QBRs, Steering Committees, Tactical Syncs) and for ad-hoc escalations related to business, project, or relationship matters.
* **Architects & Technical Leads:** Available for consultation during the design and planning phases of Joint Projects and for escalated, complex troubleshooting sessions that require deep technical expertise.
* **Engineers & Technologists:** Available for collaborative work sessions (e.g., integration, debugging, testing) as scheduled and defined within a Joint Project plan.

**Disclaimer on Impact of Unavailability:** Both parties acknowledge that the timely execution of Joint Projects and the resolution of complex issues are dependent on the mutual and reasonable availability of the personnel listed above. A lack of timely availability from either party is recognized as a project risk and may result in delays to agreed-upon project schedules and delivery timelines. The responsible Project Managers shall document any such delays and their impact, initiating a Change Request through the Joint Change Control process if necessary.

**2.3. General Communications Protocols (Human-to-Human)**

* **Incident Management:** The Provider shall maintain a public **Status Page**. For Severity 1/2 incidents, the Provider will proactively notify the Consumer's **On-Call Engineering Contact**.
* **Scheduled Maintenance:** The Provider shall announce all scheduled maintenance via the Status Page and email the **Primary Technical Contact** at least **72 hours in advance**.
* **Lifecycle Announcements:** All versioning and retirement notices will be sent via email to Primary **Technical Contact**.

**2.4. Subgroup Communication Protocols** To facilitate focused and efficient collaboration, the following communication protocols are established for specific functional subgroups.

**2.4.1. Engineer-to-Engineer Communication**

* **Purpose:** For deep technical collaboration, real-time debugging of complex integration issues, and clarifying the implementation details of the API and data services.
* **Key Participants:** Technical Integration Leads, Software Engineers from both organizations.
* **Primary Channels:** A shared, private Slack or Microsoft Teams channel; scheduled video calls for complex debugging.
* **Cadence:** As needed, driven by technical requirements and challenges.

**2.4.2. Support-to-Support Communication**

* **Purpose:** For managing, tracking, and resolving end-user-reported issues that may originate from or be affected by the service integration.
* **Key Participants:** Support Engineering Teams from both organizations.
* **Primary Channels:** A dedicated, shared ticket queue (e.g., via Zendesk); a direct email alias for escalations; a "hotline" for Severity 1 incidents.
* **Cadence:** As needed, driven by end-user support tickets, adhering strictly to the SLA.

**2.4.3. Management Sync**

* **Purpose:** To review the overall health of the business partnership, discuss performance against SLAs, address contractual items, and align on future strategic goals.
* **Key Participants:** Account Managers / Primary Business Contacts, with executive sponsors joining as needed.
* **Primary Channels:** Scheduled video conferences; email for agendas and follow-ups.
* **Cadence:** Quarterly Business Review (QBR); Monthly Check-in; Ad-hoc as required.

# Section 3: Integrated Program Management & Execution Framework

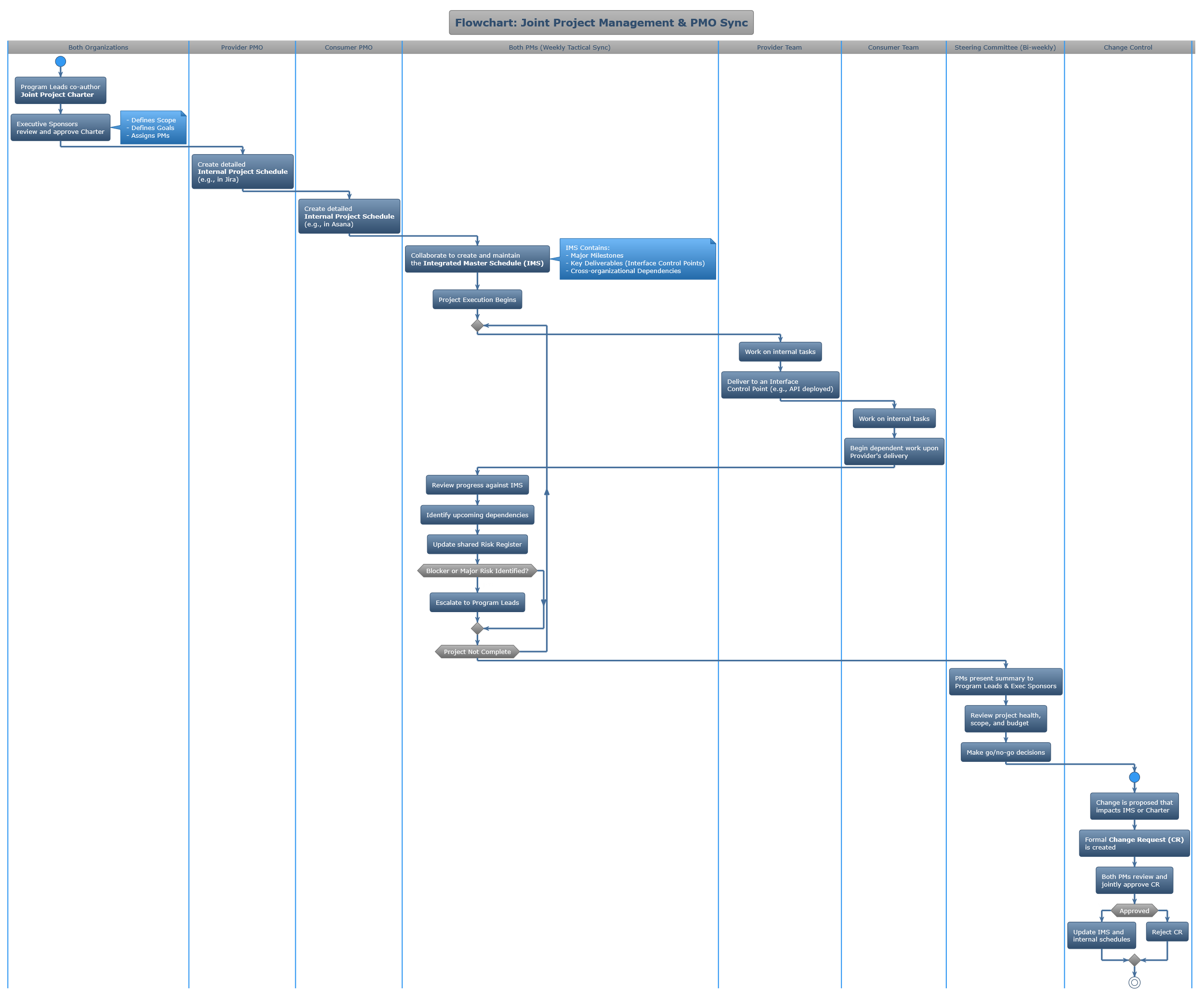
* ***Description:*** *This section establishes the formal framework for how the two organizations collaborate on and synchronize discrete, large-scale projects. It details the process for aligning on schedules, tasks, and dependencies while allowing each organization's Project Management Office (PMO) to maintain internal control over its own resources and detailed project plans.*
* ***Goal:*** *To ensure that major joint initiatives are executed in a coordinated, transparent, and predictable manner, mitigating risks, managing cross-organizational dependencies, and successfully delivering on shared objectives.*

**3.1. Scope of Applicability** This framework shall be invoked for any "Joint Project," such as the initial integration, a major API migration, or the co-development of a significant new feature.

**3.2. Program Roles and Responsibilities**

* **Executive Sponsor:** Senior leader who champions the project and serves as the final point of escalation.
* **Program Lead:** Primary business/product owner responsible for project goals.
* **Project Manager (PM):** Designated PMO contact responsible for internal planning and execution, and for synchronizing with their counterpart.

**3.3. The PMO Synchronization Process**



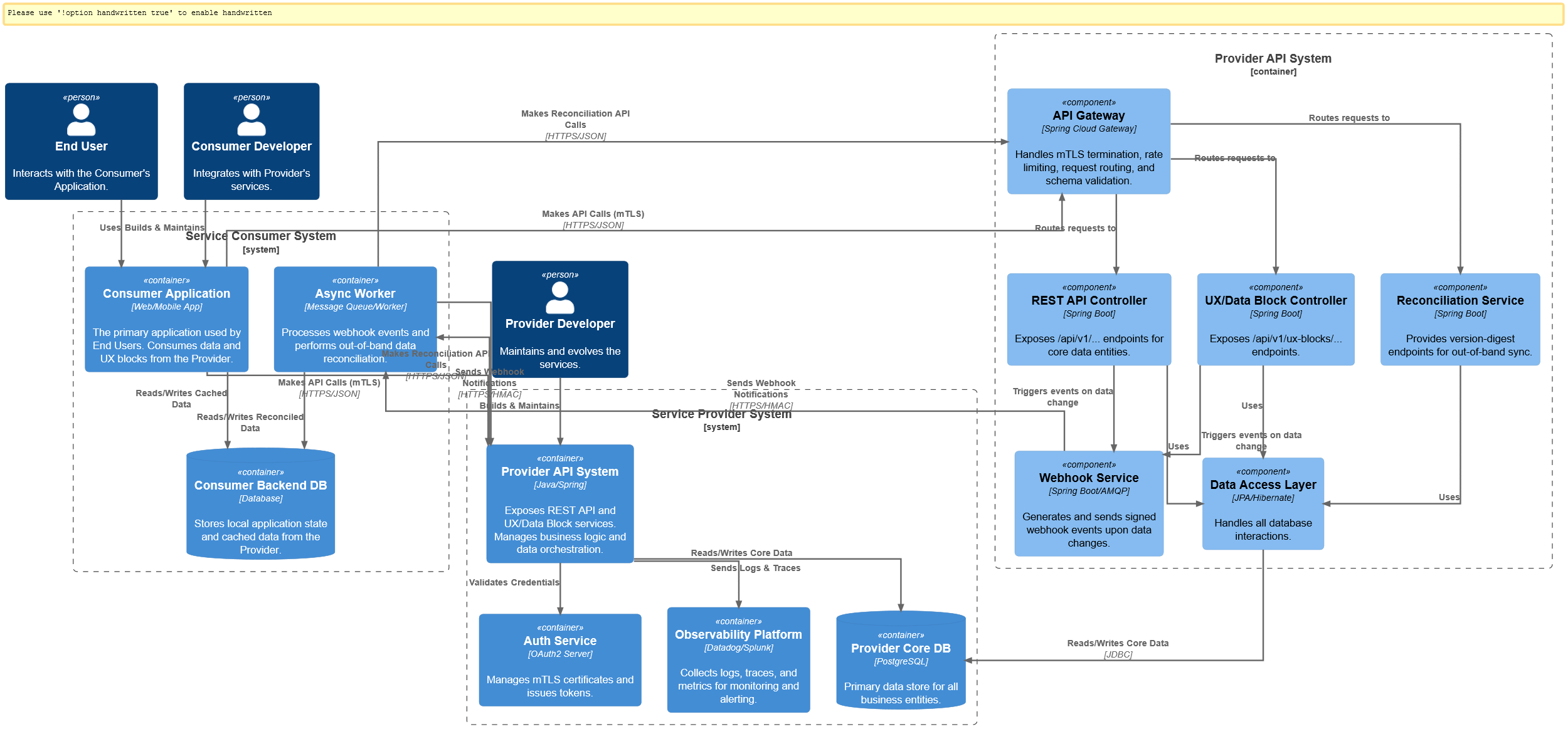
* **Step 1: Project Initiation & Chartering:** A **Joint Project Charter** is co-authored and approved, defining scope, goals, and key roles.
* **Step 2: Master Schedule Creation & Integration:** Each PM maintains a detailed internal schedule. They collaborate to create a high-level **Integrated Master Schedule (IMS)** tracking only major milestones and cross-organizational dependencies.
* **Step 3: Tasking & Dependency Management:** Tasking is managed via **Interface Control Points** (formal deliverables) in the IMS. Direct cross-organizational task assignment is prohibited; one team delivers to the interface, the other picks it up.
* **Step 4: Synchronization Cadence & Reporting:** A **Weekly Tactical Sync** for PMs to manage the IMS and risks. A **Bi-weekly/Monthly Steering Committee** for leadership to review progress and make executive decisions.
* **Step 5: Joint Change Control:** A formal **Change Request (CR)** is required for any change affecting the project's scope, schedule, or cost, ensuring mutual agreement before implementation.

**3.4. Supporting Tools & Artifacts**

* **Shared Project Management View:** A dashboard displaying the IMS.
* **Shared Document Repository:** For storing the Charter, meeting minutes, and CR log.

### Section 4: Phase II - Runtime Technical Protocol

* ***Description:*** *This section details the comprehensive technical contract for all real-time, machine-to-machine interactions. It specifies the strict protocols for security, observability, data interchange, and state management, reflecting the mature capabilities of both organizations.*
* ***Goal:*** *To ensure all data exchanges are secure, resilient, auditable, and performant by mandating specific, advanced technical standards that eliminate ambiguity and facilitate high-speed, automated operations.*

**

**4.1. General Protocol: Foundation & Security**

* **Transport & Authentication Security:**
  + **Requirement:** All server-to-server communication must be secured using **Mutual TLS (mTLS) 1.2 or higher**. This establishes a cryptographically verified identity for both the client and the server before any application data is exchanged.
  + **Certificate Management:** Both parties agree to use certificates issued by a mutually trusted Certificate Authority (CA). A joint process for annual certificate rotation will be managed via the Integrated PMO framework (Section 3), with rotation scheduled at least 30 days prior to expiration. Any emergency certificate revocation must be communicated to the counterpart's On-Call Engineering team immediately.
* **Observability & Distributed Tracing:**
  + **Requirement:** All API requests and webhook events **must** carry a distributed trace context header conforming to the **W3C Trace Context** specification.
  + **traceparent Header:** This header will be initiated by the calling service and propagated through all subsequent internal microservices on both sides of the integration. This enables end-to-end tracing of a single logical operation across both organizations.
  + **Structured Logging:** All events related to this integration (requests, responses, errors, webhook processing) must be captured in a structured JSON log format. Logs must, at a minimum, contain the trace\_id, span\_id, timestamp, event type, and relevant metadata. This allows for efficient querying and correlation of events in observability platforms (e.g., Datadog, Splunk).
* **Rate Limiting:**
  + **Requirement:** The Provider will enforce rate limits to ensure service stability. The Consumer is responsible for respecting these limits.
  + **Feedback Mechanism:** The API will provide real-time feedback via the following HTTP response headers on every request:
    - X-RateLimit-Limit: The total number of requests allowed in the current window.
    - X-RateLimit-Remaining: The number of requests remaining in the current window.
    - X-RateLimit-Reset: The Unix epoch timestamp indicating when the window resets.
  + **Policy:** When the limit is exceeded, the API will respond with an HTTP 429 Too Many Requests status code. The Consumer's client is expected to implement a backoff strategy based on the X-RateLimit-Reset header.
* **Error Handling:**
  + **Requirement:** In addition to standard HTTP status codes, all error responses (4xx, 5xx) must contain a standardized, machine-readable JSON body.
  + **Error Body Schema:**

A computer screen shot of a program

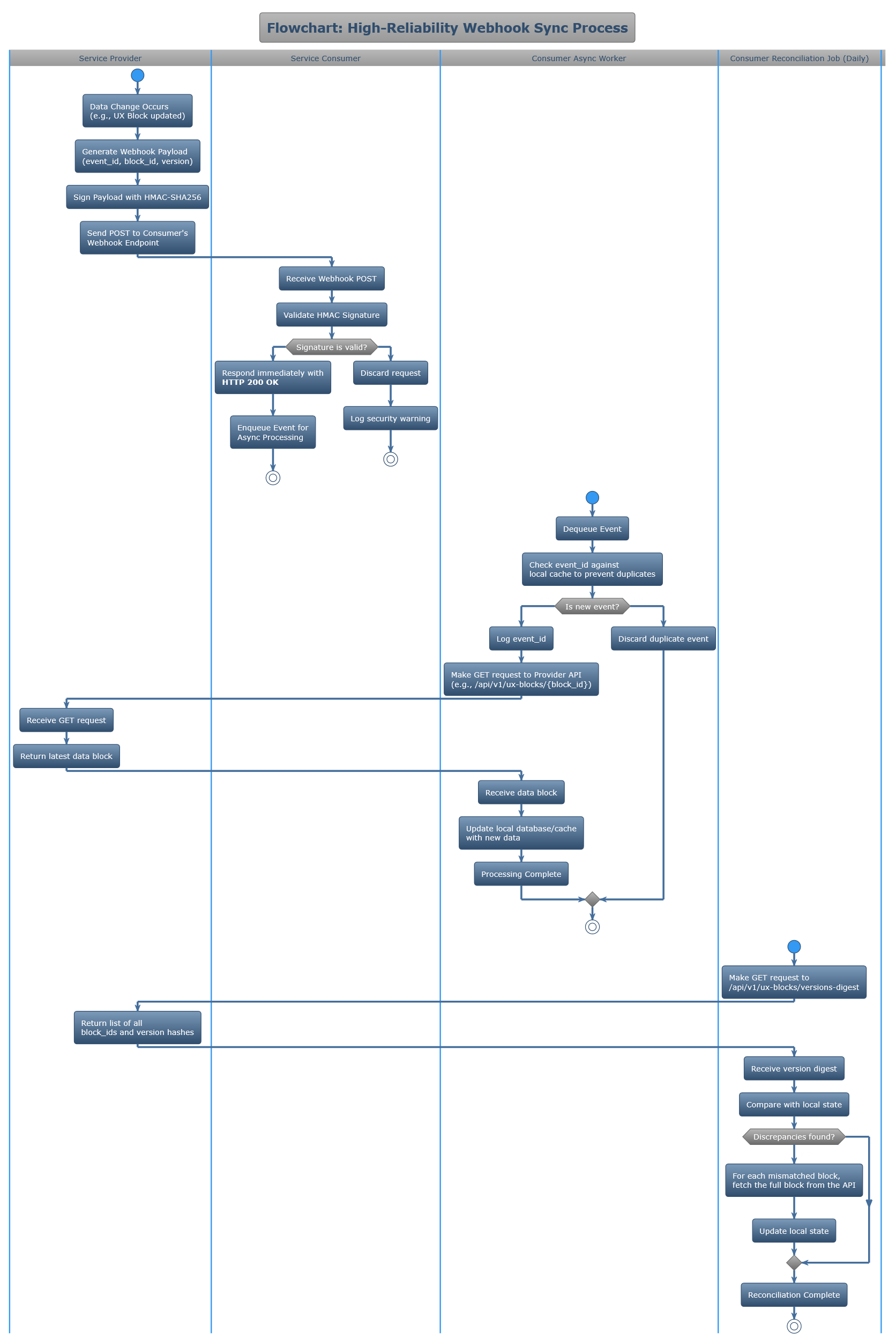
AI-generated content may be incorrect.

* + **type field:** This is an immutable, permanent URI that uniquely identifies the class of error, allowing the consumer to programmatically handle specific error types without parsing string messages.

**4.2. REST API Syncing: Data Integrity & Performance**

* **Schema Adherence:**
  + **Requirement:** All POST and PUT request bodies must strictly conform to the JSON Schema definitions published in the OpenAPI specification. The Provider's API gateway will enforce this validation. Any request with a non-conforming body will be rejected with a 400 Bad Request and a detailed invalid\_params array in the error response.
* **Advanced Caching:**
  + **Requirement:** Both parties will adhere to a formal caching strategy to minimize redundant data transfer.
  + **ETag / If-None-Match:** This must be used for polling individual resources to check for changes.
  + **Cache-Control Header:** The Provider will set appropriate Cache-Control directives (e.g., private, max-age=3600) on resource responses. The Consumer's client-side infrastructure (e.g., Redis cache) is expected to honor these directives.
* **Pagination:**
  + **Requirement:** All collection endpoints (GET /api/v1/resources) must use **cursor-based pagination** to ensure stable ordering and high performance, even when new data is being written concurrently.
  + **Mechanism:** A request will return a pagination object in the response body. To fetch the next page, the Consumer will pass the next\_cursor value in a ?cursor= query parameter. A null cursor indicates the final page has been reached.
* **Idempotency for Write Operations:**
  + **Requirement:** To prevent accidental duplicate operations, all state-changing requests (POST, PUT, DELETE) **must** include an Idempotency-Key header containing a client-generated UUID (e.g., Idempotency-Key: a8f6c6d3-2b30-4c3d-9a96-2a7f53a48e77).
  + **Mechanism:** If the Provider receives a request with a key it has already processed within the last 24 hours, it will not re-process the request and will instead return the original, cached response for that operation.

**4.3. UX/Data Block Syncing: High-Reliability Webhooks**



* **Security & Verification:**
  + **Requirement:** Webhook security will be multi-layered.
    1. **HMAC Signature:** All webhooks will be signed using HMAC-SHA256 with a pre-shared secret, delivered in an X-Provider-Signature header.
    2. **IP Allow-listing:** The Consumer must provide a list of their ingress IP addresses. The Provider will only send webhooks to these addresses, and the Consumer should only accept webhook traffic from the Provider's published egress IPs.
    3. **Timestamp Validation:** The webhook payload will include a timestamp. The Consumer must reject any webhook received more than 5 minutes after its generation time to prevent replay attacks.
* **Reliability & Delivery Guarantees:**
  + **Provider Responsibility (At-Least-Once Delivery):** If the Provider does not receive a 2xx response from the Consumer's endpoint within 3 seconds, it will consider the delivery failed. The Provider will retry delivery using an exponential backoff schedule for up to 24 hours (e.g., 5s, 30s, 5m, 30m, 2h, ...).
  + **Consumer Responsibility (Idempotent Processing):** The Consumer's endpoint **must** respond with a 200 OK immediately upon validating the webhook signature. The actual data fetching and processing must be handled asynchronously by a background worker. The Consumer must log the event\_id from the payload to de-duplicate and prevent reprocessing of retried events.
* **Out-of-Band Reconciliation:**
  + **Requirement:** To ensure state consistency in the rare event of a prolonged delivery failure or a consumer-side incident, the Consumer's system must perform a daily reconciliation.
  + **Mechanism:** The Consumer will call a dedicated REST endpoint (GET /api/v1/ux-blocks/versions-digest) which returns a list of all block\_ids and their current version hashes. The Consumer will compare this digest against its local state and fetch any blocks that are missing or have a mismatched version hash.

### Section 5: Phase III - Service Evolution & Versioning

* ***Description:*** *This section provides a rigorous, collaborative framework for managing the service lifecycle. It details the precise process for introducing both breaking and non-breaking changes, designed for mature organizations that need to co-plan significant architectural shifts.*
* ***Goal:*** *To enable continuous improvement and innovation in the Provider's services while affording the Consumer maximum predictability, participation, and lead time to adapt, thereby protecting both businesses from unplanned disruption.*

**5.1. Versioning Philosophy & Scheme**

* **Requirement:** All services, including the REST API and the schema of UX/Data Blocks, must strictly adhere to **Semantic Versioning (MAJOR.MINOR.PATCH)**. The versioning contract applies to the API surface (endpoints, parameters), data schemas, and documented functional behavior.

**5.2. Process for Introducing a Breaking (MAJOR) Change**

This process is designed as a joint program of work managed under the Integrated PMO Framework (Section 3).

* **T-minus 7 Months (Joint Design Review):** The Provider's architects will present the proposed technical design and draft OpenAPI specification for the new version (v2) to the Consumer's architects. This collaborative session is intended to identify any fundamental concerns or major integration challenges from the Consumer's perspective before development begins.
* **T-minus 6 Months (Formal Announcement):** Following feedback incorporation, the Provider will make the formal deprecation announcement for v1 and publish the detailed v2 specification and a comprehensive migration guide. A dedicated, shared Slack channel for the migration project will be created.
* **T-minus 3 Months (Sandbox Deployment):** v2 is deployed to the Sandbox environment, functionally complete. Both parties will engage in joint integration testing.
* **T-minus 1 Month (Production Parallel Deployment):** v2 is deployed to Production. Both v1 and v2 will run in parallel and be fully supported. The Provider will supply a shared metrics dashboard showing the Consumer's traffic volume to both v1 and v2 endpoints, allowing the Consumer to verify their migration progress in real time.
* **Launch Date (Default Switch):** v2 becomes the default version for any new consumers. v1 is now officially "deprecated."
* **Launch Date + 6 Months (Sunset):** The v1 endpoints are fully retired as per the process in Section 7.

**5.3. Process for Non-Breaking (MINOR/PATCH) Changes**

Even non-breaking changes will follow a transparent and predictable release process.

* **Communication:** All upcoming MINOR and PATCH changes will be documented in a shared, forward-looking release notes document accessible to the Consumer's technical team. MINOR changes will be formally announced to the Primary Technical Contact at least **2 weeks** prior to production deployment.
* **Pre-release Testing (Sandbox Opt-in):** At least **1 week** before production deployment, all MINOR/PATCH changes will be deployed to the Sandbox environment. Mature consumers can opt-in to test these pre-release versions by sending a custom Accept header (e.g., Accept: application/vnd.provider.v1.2-beta+json) to route their Sandbox requests to the new version.
* **Deployment:** Production deployments of non-breaking changes will occur during pre-announced, low-impact maintenance windows. A detailed, public-facing changelog will be updated immediately upon deployment.

### Section 6: Phase IV - Service Retirement & Sunsetting

* ***Description:*** *This section provides a comprehensive, rigorous, and transparent framework for the final stage of a service's lifecycle. It details the precise procedures, communication protocols, and technical measures for retiring a service or a deprecated MAJOR version of a service.*
* ***Goal:*** *To ensure that the retirement process is executed as a planned, joint program of work, providing the Consumer with maximum predictability and extensive lead time to migrate. This framework is designed to eliminate abrupt disruptions, protect the Consumer's business operations, and facilitate a smooth transition to successor services.*

**6.1. Principle & Scope of Applicability**

* **Guiding Principle:** Service retirement is not an operational failure but a planned evolution. This process is founded on the principles of proactive communication, mutual cooperation, and a shared commitment to a non-disruptive transition.
* **Scope of Applicability:** The procedures, timelines, and technical requirements outlined in this section apply equally to two distinct scenarios:
  1. **Complete Service Retirement:** The permanent discontinuation of an entire service offering (e.g., retiring the "UX/Data Block Service").
  2. **MAJOR Version Sunsetting:** The permanent discontinuation of a deprecated MAJOR version of a service (e.g., sunsetting v1 of the REST API after v2 has become the standard).

**6.2. The Deprecation Period**

Upon formal announcement, the service or version enters a "deprecated" state for a predefined period.

* **Minimum Duration:**
  + For **Complete Service Retirement**: A minimum of **12 months**.
  + For **MAJOR Version Sunsetting**: A minimum of **6 months**, commencing from the "Launch Date" defined in Section 5.2.
* **Service Level Guarantees during Deprecation:**
  + **SLA Adherence:** The service level guarantees for API Availability and Performance as defined in Section 2.2.1 **shall remain in full effect** throughout the deprecation period.
  + **Feature Freeze:** The deprecated service/version is considered feature-frozen. No new MINOR or PATCH features will be developed or deployed.
  + **Maintenance:** Maintenance will be limited to the implementation of critical security patches and bug fixes necessary to prevent a Severity 1 or Severity 2 incident.
* **Prohibition of New Usage:** During the deprecation period, the Consumer organization is prohibited from initiating new development or integrations against the deprecated service/version. All new work must target the designated successor service.

**6.3. Communication Protocol during Deprecation**

A multi-channel communication plan will be executed to ensure constant awareness.

* **Formal Announcements:** A schedule of formal written notices will be sent to the Consumer's **Primary Technical Contact** and **Account Manager** at the following intervals prior to the final shutdown date:
  + Initial Announcement (Start of Period)
  + 3-Month Reminder
  + 1-Month Reminder
  + 1-Week Reminder
  + 24-Hour Final Notice
* **Governance Reporting:** During the deprecation period, a "Deprecation Status" section will be a mandatory agenda item in all Quarterly Business Reviews (QBRs). This report will include metrics on the Consumer's remaining traffic volume to the deprecated endpoints, allowing both parties to track migration progress against established goals.
* **Status Page:** The affected service/version will be clearly marked as "Deprecated" with the scheduled retirement date prominently displayed on the Provider's public status page.

**6.4. Technical Measures during Deprecation**

Specific technical signals will be implemented to facilitate automated and manual discovery of dependencies.

* **API Response Headers:** For the entire duration of the deprecation period, every response from the deprecated endpoint(s) **must** include the following HTTP headers, conforming to RFC 8594:
  + Deprecation: Sat, 25 Jan 2026 15:00:00 GMT *(Note: The date used is the planned shutdown date)*
  + Link: <https://developer.provider.com/api/v2>; rel="successor-version"
* **Planned "Brownouts":** In the final month of the deprecation period, the Provider reserves the right to institute scheduled, temporary, and intentional outages known as "brownouts."
  + **Purpose:** To assist the Consumer in identifying any remaining, undiscovered, or non-obvious dependencies on the deprecated service before the final shutdown.
  + **Scheduling & Notification:** Brownouts will be brief (5-15 minutes), occur during pre-defined low-traffic maintenance windows, and will be announced to the **On-Call Engineering Contact** at least 48 hours in advance.
  + **Technical Behavior:** During a brownout, the service will return an HTTP 503 Service Unavailable status code with a response body clearly indicating that this is a planned deprecation brownout exercise.

**6.5. Final Shutdown & Post-Retirement State**

* **The Shutdown Event:** At the exact time specified in the final notice, all network traffic to the retired endpoints will be terminated at the Provider's edge load balancer.
* **Post-Retirement Response:**
  + **Requirement:** To prevent client-side timeouts and provide a clear, final status, the retired endpoints **must not** be removed from the network configuration. For a period of at least **12 months post-retirement**, any request to a retired endpoint must immediately return an HTTP 410 Gone status code.
  + **Response Body:** The body of the 410 Gone response will be a JSON object containing a link to the successor service's documentation and the request's trace\_id for logging.
* **Data & Resource Archival:** All data exclusively associated with the retired service or version will be managed according to the master data retention and archival policy mutually agreed upon in the master commercial contract between the two organizations.

### Section 7: Phase IV - Service Retirement & Sunsetting

* **Description:** This section provides a comprehensive, rigorous, and transparent framework for the final stage of a service's lifecycle. It details the precise procedures, communication protocols, and technical measures for retiring a service or a deprecated MAJOR version of a service.
* **Goal:** To ensure that the retirement process is executed as a planned, joint program of work, providing the Consumer with maximum predictability and extensive lead time to migrate. This framework is designed to eliminate abrupt disruptions, protect the Consumer's business operations, and facilitate a smooth transition to successor services.

As this agreement marks the commencement of a new initiative, both parties acknowledge that the immediate need and risk associated with a strict sunsetting or retirement process are presently low. Any such event is projected to be far in the future. It is therefore anticipated that this agreement, including the specifics of this retirement protocol, will be amended and refined over time through practical use, joint operational experience, and the adoption of updated technological approaches. Nevertheless, the following detailed process is established herein as the formal, governing procedure to ensure long-term predictability for both organizations.

**7.1. Principle & Scope of Applicability**

* **Guiding Principle:** Service retirement is not an operational failure but a planned evolution. This process is founded on the principles of proactive communication, mutual cooperation, and a shared commitment to a non-disruptive transition.
* **Scope of Applicability:** The procedures, timelines, and technical requirements outlined in this section apply equally to two distinct scenarios:
  1. **Complete Service Retirement:** The permanent discontinuation of an entire service offering (e.g., retiring the "UX/Data Block Service").
  2. **MAJOR Version Sunsetting:** The permanent discontinuation of a deprecated MAJOR version of a service (e.g., sunsetting v1 of the REST API after v2 has become the standard).

**7.2. The Deprecation Period**

Upon formal announcement, the service or version enters a "deprecated" state for a predefined period.

* **Minimum Duration:**
  + For **Complete Service Retirement**: A minimum of **12 months**.
  + For **MAJOR Version Sunsetting**: A minimum of **6 months**, commencing from the "Launch Date" defined in Section 6.
* **Service Level Guarantees during Deprecation:**
  + **SLA Adherence:** The service level guarantees for API Availability and Performance as defined in Section 2.2.1 **shall remain in full effect** throughout the deprecation period.
  + **Feature Freeze:** The deprecated service/version is considered feature-frozen. No new MINOR or PATCH features will be developed or deployed.
  + **Maintenance:** Maintenance will be limited to the implementation of critical security patches and bug fixes necessary to prevent a Severity 1 or Severity 2 incident.
* **Prohibition of New Usage:** During the deprecation period, the Consumer organization is prohibited from initiating new development or integrations against the deprecated service/version. All new work must target the designated successor service.

**7.3. Communication Protocol during Deprecation**

A multi-channel communication plan will be executed to ensure constant awareness.

* **Formal Announcements:** A schedule of formal written notices will be sent to the Consumer's **Primary Technical Contact** and **Account Manager** at the following intervals prior to the final shutdown date:
  + Initial Announcement (Start of Period)
  + 3-Month Reminder
  + 1-Month Reminder
  + 1-Week Reminder
  + 24-Hour Final Notice
* **Governance Reporting:** During the deprecation period, a "Deprecation Status" section will be a mandatory agenda item in all Quarterly Business Reviews (QBRs). This report will include metrics on the Consumer's remaining traffic volume to the deprecated endpoints, allowing both parties to track migration progress against established goals.
* **Status Page:** The affected service/version will be clearly marked as "Deprecated" with the scheduled retirement date prominently displayed on the Provider's public status page.

**7.4. Technical Measures during Deprecation**

Specific technical signals will be implemented to facilitate automated and manual discovery of dependencies.

* **API Response Headers:** For the entire duration of the deprecation period, every response from the deprecated endpoint(s) **must** include the following HTTP headers, conforming to RFC 8594:
  + Deprecation: Sun, 26 Jan 2026 15:10:50 GMT *(Note: Example date is 6 months from the effective date)*
  + Link: <https://developer.provider.com/api/v2>; rel="successor-version"
* **Planned "Brownouts":** In the final month of the deprecation period, the Provider reserves the right to institute scheduled, temporary, and intentional outages known as "brownouts."
  + **Purpose:** To assist the Consumer in identifying any remaining, undiscovered, or non-obvious dependencies on the deprecated service before the final shutdown.
  + **Scheduling & Notification:** Brownouts will be brief (5-15 minutes), occur during pre-defined low-traffic maintenance windows, and will be announced to the **On-Call Engineering Contact** at least 48 hours in advance.
  + **Technical Behavior:** During a brownout, the service will return an HTTP 503 Service Unavailable status code with a response body clearly indicating that this is a planned deprecation brownout exercise.

**7.5. Final Shutdown & Post-Retirement State**

* **The Shutdown Event:** At the exact time specified in the final notice, all network traffic to the retired endpoints will be terminated at the Provider's edge load balancer.
* **Post-Retirement Response:**
  + **Requirement:** To prevent client-side timeouts and provide a clear, final status, the retired endpoints **must not** be removed from the network configuration. For a period of at least **12 months post-retirement**, any request to a retired endpoint must immediately return an HTTP 410 Gone status code.
  + **Response Body:** The body of the 410 Gone response will be a JSON object containing a link to the successor service's documentation and the request's trace\_id for logging.
* **Data & Resource Archival:** All data exclusively associated with the retired service or version will be managed according to the master data retention and archival policy mutually agreed upon in the master commercial contract between the two organizations.

# Section 8: Agreement & Signatures

* ***Description:*** *This final section serves as the formal execution of this agreement.*
* ***Goal:*** *To create a binding commitment from both parties to adhere to all terms, conditions, and protocols laid out in this document.*

By signing below, the authorized representatives of the Service Provider and Service Consumer acknowledge that they have read, understood, and agree to be bound by the terms of this agreement.

**For the Service Provider:**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**For the Service Consumer:**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Title: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_