NGX-F215343: Rx Enabler (Arch) – Correct RX Access for Carved-out Members

# Feature Summary

Objective: Prevent display or use of pharmacy claims and prescriptions that fall outside a member’s Aetna-covered period when the member’s pharmacy history originates from CVS Caremark. CVS systems legitimately maintain a longitudinal (lifetime) prescription history for an individual, spanning time periods when the person may have had other insurers or no coverage at Aetna. When Aetna consumes CVS data via the Consolidated Claims and related APIs, those lifetime records can include fills that pre‑date (and in some cases post‑date) the member’s Aetna plan effective dates. If unfiltered, these records would be visible in Aetna Health and could be acted upon by downstream services.

Solution: Introduce a Policy Effective Window Filter that gates visibility and use of any CVS‑sourced prescription claim, order, or refill. The filter constrains records to the intersection of: (a) the member’s pharmacy membership effective window that aligns with the Aetna plan effective dates, (b) the client’s requested startDate/endDate, and (c) attestation‑controlled integration state. Only claims with fillDate within this window are eligible for display and subsequent flows (detail, order, refill). The integration itself is governed by the ‘CVSRxCarveOut’ feature toggle/attestation; when enabled, Aetna presents embedded capabilities and SSO; when disabled, Aetna presents a limited, link‑out experience.

# Discovery

End‑to‑end prescription management touches multiple services and rules. The discovery synthesized the following dependencies, implications, and constraints that must be addressed uniformly to avoid inconsistent behavior:

1) Data & Authorization Dependencies:  
 • CVS Auth (patient profile) issues a token used by Aetna to call CVS backends.  
 • Consolidated Claims (/getClaimsHistory) returns all visible plan‑member claims for the authenticated member.  
 • Core Proxy (/userauthorizations) and Member Profile (/familyaccessrules) supply authorization and dependent visibility.  
 • UAF privacy checks further restrict access (e.g., HIPAA privacy restrictions, sensitive medications).

2) Filtering & Normalization Dependencies:  
 • Policy Effective Window Filter uses pharmacy membership effectiveStart/effectiveEnd to bound results.  
 • Request window validation requires startDate and endDate; errors if missing.  
 • Deduplicate by uniqueRxId and retain the latest fillDate for list responses.  
 • Final restriction by membershipResourceIds if provided.

3) Integration Surface Area:  
 • Five key integrations (PLP, PDB, PlaceOrder, Ship Consent, i90) depend on consistent claim visibility.  
 • ~47 related APIs rely on the same underlying data and rules; front‑end logic cannot diverge.  
 • A crosswalk/mapping layer is required to align identifiers and payloads across systems.

4) Behavioral Edge Cases to Handle:  
 • Claims with fillDate before plan effectiveStart: exclude.  
 • Orders/refills that straddle coverage boundaries: exclude or block action if outside window.  
 • RefillsLeft = 0 or missing: ensure accurate messaging and no spurious eligibility.  
 • Attestation off (no integrated carve‑out): present limited, link‑out experience.  
 • Missing dates or invalid ranges: return 400 with actionable error details.

5) Caching, Observability, and Security:  
 • Cache key includes account idSource~idValue, membershipResourceId, startDate, endDate.  
 • Log decision points (authorization, privacy, effective window trimming, dedup) for auditability.  
 • Enforce HIPAA and PHI handling standards; secure token exchange and storage.

# Options

## Option 1: Centralized Eligibility & Display Logic Service

Centralize claim visibility rules in a middleware service that evaluates attestation flags, plan effective dates, and carve‑out status before returning data to clients. This ensures consistency across all integrations (PLP, PDB, PlaceOrder, Ship Consent, i90) and eliminates undocumented front‑end logic.

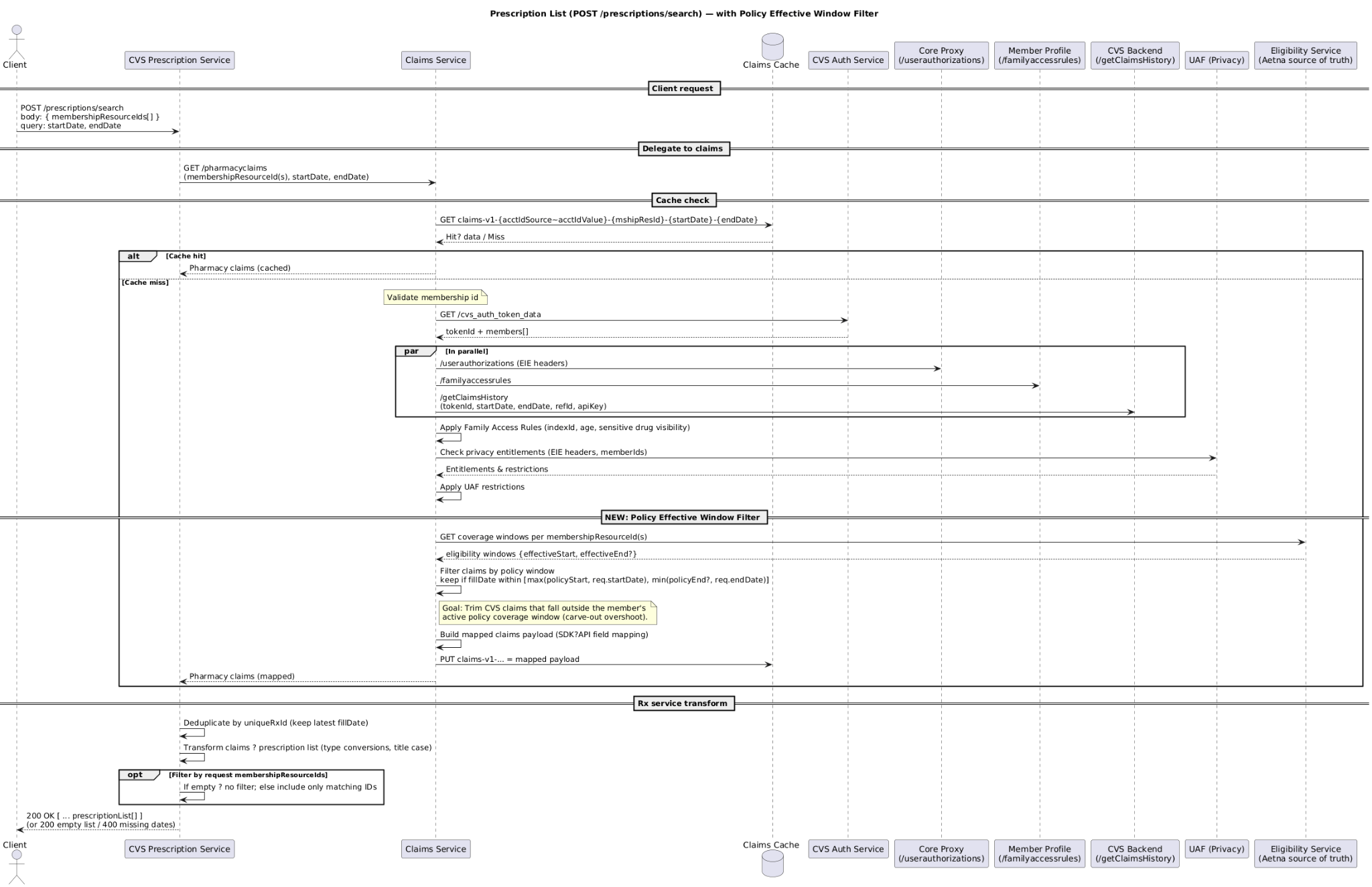
## Option 2: Distributed Logic in Clients

Embed a shared validation library in each integration to enforce claim visibility rules locally. This offers independence but risks divergence over time.

Selected: Option 1 – Centralized service approach for scalability and compliance.

# Solution Sketch

The following diagram illustrates the updated Prescription List Flow with Effective Date Policy Filtering applied. It replaces the previous orchestration placeholder and demonstrates where effective date filtering is enforced.



# Service Logic

The centralized eligibility service enforces claim visibility rules in this order:  
1) Validate attestation (CVSRxCarveOut) and integration state.  
2) Retrieve CVS token (patient profile), authorizations, family access rules, and privacy entitlements.  
3) Fetch claims and normalize; deduplicate by uniqueRxId keeping the latest fillDate.  
4) Apply Policy Effective Window Filter using pharmacy membership effective dates ∩ request window.  
5) Apply family access/UAF privacy restrictions.  
6) Restrict by membershipResourceIds if provided and return mapped responses.

# Services

• CVS Auth Service (patient profile token)  
• CVS Consolidated Claims Service  
• Prescription List Service (POST /prescriptions/search; GET /prescriptions/detail)  
• Order & Refill Service  
• Core Proxy (/userauthorizations)  
• Member Profile (/familyaccessrules)  
• UAF (privacy)  
• Centralized Eligibility Middleware (Effective Date & Attestation Filtering)

# Non-Functional Requirements (NFR)

Performance: Response ≤ 200ms under normal load.  
Scalability: Support ~47 APIs and five key integrations.  
Caching: Use deterministic cache keys and TTL aligned with CVS freshness.  
Observability: Emit metrics and traces for cache hits, trims by window, privacy filters, and dedup.  
Reliability: Graceful degradation when dependencies are unavailable.

# Scenarios

1) Prescription claim before coverage effectiveStart → excluded from all views and actions.  
2) Refill/order outside coverage window → blocked with clear messaging.  
3) Member without signed attestation → link‑out (no embedded capabilities).  
4) Member with signed attestation → full embedded integration.  
5) Missing startDate/endDate → 400 Bad Request with guidance.  
6) Sensitive drug + privacy restriction → excluded per UAF.

# Security Review

• All calls use TLS; tokens stored and rotated per policy.  
• PHI/PII minimized, masked in logs; privacy decisions audited.  
• Access governed by EIE headers, family access, and UAF entitlements.  
• Regular reviews with Security for control effectiveness.

# Client Changes

• Integrate with centralized eligibility service for consistent filtering.  
• Enforce and display messaging when items are hidden due to effective dates or privacy.  
• Respect attestation state to show embedded features vs link‑out.  
• Keep UI and mobile parity across platforms.

# Services (Expanded)

Below are the core services and their interactions, expanded with field-level mappings and sample payloads to ensure precise architectural alignment.  
  
• CVS Auth Service (`GET /cvs\_auth\_token\_data`): issues token with member demographics and identifiers.  
• Consolidated Claims Service (`GET /pharmacyclaims`): retrieves raw claim history across visible plan members.  
• Prescription List Service (`POST /prescriptions/search`): applies filtering, deduplication, effective date window.  
• Prescription Detail Service (`GET /prescriptions?claimId=…`): enriches a single prescription with dosage, refills, cost.  
• Order & Refill Service: manages refill eligibility and order placement with effective date enforcement.  
• Family Access Rules (`/familyaccessrules`): ensures dependent entitlements are respected.  
• UAF Service: privacy restrictions enforced per member and medication.  
• Centralized Eligibility Middleware: authoritative Policy Effective Window enforcement.

# Field Mapping Tables

Prescription List (POST /prescriptions/search):  
  
• .details[].memberClaims[].claimNumber → prescriptionList[].claimId  
• .details[].memberClaims[].uniqueRxId → prescriptionList[].prescriptionId  
• .details[].memberClaims[].drug.name → prescriptionList[].drugName (Title Case)  
• .details[].memberClaims[].drug.strength → prescriptionList[].drugStrength (String→Number)  
• .details[].memberClaims[].daysSupplyQuantity → prescriptionList[].daysSupply (String→Number)  
• .details[].memberClaims[].dispensedQuantity → prescriptionList[].quantity (String→Number)  
• .details[].memberFirstName → memberFirstName  
• .details[].memberLastName → memberLastName  
• .details[].memberClaims[].fillDate → prescriptionList[].lastFilledDate (Dedup: keep latest fillDate)  
• .details[].memberClaims[].prescriberName → prescriptionList[].prescriberName  
• .details[].memberClaims[].pharmacyName → prescriptionList[].fulfilledBy  
• .details[].relationshipToSubscriber → relationshipToSubscriber  
• .details[].membershipResourceId → membershipResourceId  
• prescriptionsRequest.startDate → startDate  
• prescriptionsRequest.endDate → endDate

Prescription Detail (GET /prescriptions?claimId=…):  
  
• .details[].memberClaims[].claimNumber → claimId  
• .details[].memberClaims[].uniqueRxId → prescriptionId  
• .details[].memberClaims[].fillDate → lastFilledDate  
• .details[].memberFirstName → memberFirstName  
• .details[].memberLastName → memberLastName  
• .details[].relationshipToSubscriber → relationshipToSubscriber  
• .details[].membershipResourceId → membershipResourceId  
• .details[].memberClaims[].drug.name → drugName  
• .details[].memberClaims[].drug.strength → drugStrength  
• .details[].memberClaims[].drug.dosageForm → drugForm  
• .details[].memberClaims[].daysSupplyQuantity → daysSupply  
• .details[].memberClaims[].dispensedQuantity → quantity  
• .details[].memberClaims[].refillsLeft → refillsLeft  
• .details[].memberClaims[].prescriber.fullName → prescriberName  
• .details[].memberClaims[].pharmacyName → fulfilledBy  
• (.memberClaims[].payAmount + .clientPayAmount) → estimatedCost  
• .details[].memberClaims[].rxNumber → prescriptionNumber  
• .details[].memberClaims[].drug.ndcId → NDC11

# Example Payloads

Prescription List Response (truncated example):  
  
[  
 {  
 "membershipResourceId": "5~263801696+31+1+20180101+788678+C+3",  
 "memberFirstName": "John",  
 "memberLastName": "Doe",  
 "relationshipToSubscriber": "Self",  
 "startDate": "2019-04-01",  
 "endDate": "2019-12-05",  
 "prescriptionList": [  
 {  
 "claimId": "200023611694001",  
 "prescriptionId": "674529584",  
 "drugName": "Lipitor",  
 "drugStrength": "200mg",  
 "quantity": 15,  
 "lastFilledDate": "2018-04-12",  
 "daysSupply": 5,  
 "fulfilledBy": "Caremark prescription service",  
 "prescriberName": "John Doe"  
 }  
 ]  
 }  
]

Prescription Detail Response (truncated example):  
  
{  
 "claimId": "200023611694001",  
 "prescriptionId": "674529584",  
 "drugName": "Lipitor",  
 "drugStrength": "200mg",  
 "drugForm": "Tablet",  
 "daysSupply": 5,  
 "quantity": 15,  
 "lastFilledDate": "2018-04-12",  
 "refillsLeft": 0,  
 "prescriberName": "John Doe",  
 "fulfilledBy": "Caremark prescription service",  
 "estimatedCost": 111,  
 "prescriptionExpirationDate": "2018-04-12",  
 "prescriptionNumber": "674529584",  
 "NDC11": "536012297"  
}

# Scenarios

1) Prescription claim with fillDate before coverage effectiveStart → excluded from both list and detail.  
2) Refill order straddling effective window (e.g., refill date before effectiveStart, claim date within) → excluded.  
3) Refill request with refillsLeft = 0 → disallowed with clear message.  
4) Member without attestation (CVSRxCarveOut = false) → UI shows link-out, not embedded services.  
5) Missing startDate/endDate query parameters → service throws 400 Bad Request.  
6) Claims flagged as sensitive (isSensitive=true) → excluded if UAF rules deny visibility.  
7) Dependent under 18 without family access rules → excluded unless explicitly authorized.  
8) Duplicate claims with same uniqueRxId → keep only claim with latest fillDate.

# API & Swagger (Pharmacy Service v1)

## Overview

This section defines the authoritative API contract for the Pharmacy Service enforcing the Policy Effective Window Filter across list, detail, and generic visibility use cases. All routes are prefixed with /v1. The full OpenAPI 3.0.3 specification is attached separately; this section provides a concise summary and examples.

Endpoints (v1):  
• GET /v1/health – service probe  
• GET /v1/coverage/windows – resolve Aetna-aligned effective coverage windows for one or more members  
• POST /v1/prescriptions/search – authoritative list (window trim, privacy/family access, dedup)  
• GET /v1/prescriptions/{claimId} – authoritative detail view  
• POST /v1/visibility/evaluate – generic batch evaluator for new/adjacent use cases

## Versioning & Stability

All endpoints are versioned under /v1. Breaking changes will result in a new major prefix (/v2). Non-breaking, additive fields are gated via tolerant readers and do not change the version path.

## Error Model & Failure IDs

Responses are wrapped in a standard envelope. For errors (4xx/5xx), the API returns an ErrorEnvelope with additional\_data carrying canonical failure identifiers and reasons. These are suitable for telemetry and client-side branching.

Common failure\_id values:  
• MISSING\_REQUIRED\_DATE – startDate and/or endDate missing  
• INVALID\_DATE\_FORMAT – dates not ISO YYYY-MM-DD  
• INVALID\_DATE\_RANGE – startDate after endDate  
• NOT\_ATTESTED – embedded capabilities disallowed by attestation  
• CLAIM\_NOT\_FOUND – claimId not associated to visible member  
• CLAIM\_OUTSIDE\_EFFECTIVE\_WINDOW – claim exists but fillDate outside effective window  
• HIDDEN\_BY\_PRIVACY / HIDDEN\_BY\_FAMILY\_ACCESS – filtered by UAF/family rules  
• INVALID\_MEMBERSHIP\_ID – malformed membershipResourceId  
• TOKEN\_EXPIRED / UPSTREAM\_UNAVAILABLE – dependency failures

## Partial Results Semantics (200 vs 206)

When a list/batch contains a mix of eligible and ineligible items, the API does not fail the request. By default it returns HTTP 200 with additional\_data.code=PARTIAL\_SUCCESS, placing only eligible items in data and capturing excluded items under additional\_data.partial\_failures[].

Clients can opt into explicit partial semantics via either:  
• Header: Prefer: partial=206  
• Query: partialAs=206  
When used, the service responds with HTTP 206 and includes headers X-Partial-Result=true and X-Partial-Reason.

## Brief Examples

POST /v1/prescriptions/search?startDate=2019-01-01&endDate=2019-12-31

Request body (subset):

{  
 "membershipResourceIds": ["5~263801696+31+1+20180101+788678+C+3"],  
 "options": {"applyPrivacy": true, "applyFamilyAccess": true, "applyDeduplication": true, "requireAttestation": true}  
}

Response 200 (partial success, truncated):

{  
 "statusCode": 200,  
 "statusDescription": "OK",  
 "additional\_data": {  
 "code": "PARTIAL\_SUCCESS",  
 "partial": true,  
 "counts": {"total": 2, "eligible": 1, "hiddenByEffectiveWindow": 1},  
 "partial\_failures": [{  
 "failure\_id": "CLAIM\_OUTSIDE\_EFFECTIVE\_WINDOW",  
 "diagnostics": {"claimId": "200023611694002", "fillDate": "2018-04-12"}  
 }]  
 },  
 "data": [{  
 "membershipResourceId": "5~263801696+31+1+20180101+788678+C+3",  
 "prescriptionList": [{"claimId": "200023611694001", "drugName": "Lipitor", "lastFilledDate": "2019-08-12"}]  
 }]  
}

POST /v1/visibility/evaluate?startDate=2019-01-01&endDate=2019-12-31&partialAs=206 (client opts into 206)

Response 206 (partial content, truncated):

{  
 "statusCode": 200,  
 "statusDescription": "OK",  
 "additional\_data": {"code": "PARTIAL\_SUCCESS", "partial": true},  
 "data": {"decisions": [{"claimNumber": "200023611694001", "visibility": "eligible"}]}  
}  
Headers: X-Partial-Result: true; X-Partial-Reason: CLAIM\_OUTSIDE\_EFFECTIVE\_WINDOW

## Artifacts

• OpenAPI 3.0.3: pharmacy-service-v1d.yaml

• Postman: pharmacy-service.postman\_collection.v2.json, pharmacy-service.postman\_environment.json

• Client stubs: pharmacy-service-client-stubs.ts

## Recommendations (Document Structure & Placement)

1) Move “Services (Expanded)” below this API & Swagger section so contract precedes implementation detail.  
2) Add an Appendix: “Interface Contracts & Field Mappings” with side-by-side tables (CVS → Aetna) to aid dev/QA.  
3) Add an Appendix: “Failure Catalog” enumerating failure\_id, reason, and remediation.  
4) In Scenarios, add explicit partial-success cases (200 vs 206) with UI guidance.  
5) In NFRs, add SLI/SLOs tied to this API (latency P95, error budget, partial-rate baseline).  
6) Add an Observability section: required logs/metrics/traces and correlation IDs.

# Appendix: Swagger Summary (Pharmacy Service v1)

openapi: 3.0.3  
info:  
 title: Pharmacy Service  
 version: 1.0.0  
servers:  
 - url: https://sandbox.aetnahealth.example.com  
paths:  
 /v1/prescriptions/search: POST  
 /v1/prescriptions/{claimId}: GET  
 /v1/visibility/evaluate: POST  
 /v1/coverage/windows: GET  
x-partial-handling:  
 default\_http\_status\_for\_partials: 200  
 override:  
 header: "Prefer: partial=206|200"  
 query: "partialAs=206|200"

ErrorEnvelope:  
 statusCode: integer  
 statusDescription: string  
 additional\_data:  
 failure\_id: string  
 failure\_reason: string  
 diagnostics: object  
 code: "PARTIAL\_SUCCESS"?  
 partial: boolean?  
 counts: object?  
 partial\_failures: [ { failure\_id, failure\_reason, diagnostics } ]  
 data: any | null

Example — POST /v1/prescriptions/search?startDate=2019-01-01&endDate=2019-12-31  
Request:  
{  
 "membershipResourceIds": ["5~263801696+31+1+20180101+788678+C+3"],  
 "options": {"applyPrivacy": true, "applyFamilyAccess": true, "applyDeduplication": true, "requireAttestation": true}  
}  
Response (200, PARTIAL\_SUCCESS):  
{  
 "statusCode": 200,  
 "statusDescription": "OK",  
 "additional\_data": {  
 "code": "PARTIAL\_SUCCESS", "partial": true,  
 "counts": {"total": 2, "eligible": 1, "hiddenByEffectiveWindow": 1}  
 },  
 "data": [ { "membershipResourceId": "5~...",  
 "prescriptionList": [ { "claimId": "200023611694001", "drugName": "Lipitor", "lastFilledDate": "2019-08-12" } ] } ]  
}

Example — POST /v1/visibility/evaluate?startDate=2019-01-01&endDate=2019-12-31&partialAs=206  
Response (206, headers: X-Partial-Result=true; X-Partial-Reason=CLAIM\_OUTSIDE\_EFFECTIVE\_WINDOW):  
{  
 "statusCode": 200,  
 "statusDescription": "OK",  
 "additional\_data": { "code": "PARTIAL\_SUCCESS", "partial": true },  
 "data": { "decisions": [ { "claimNumber": "200023611694001", "visibility": "eligible" } ] }  
}

## Artifact Hyperlinks

OpenAPI YAML: [pharmacy-service-v1d.yaml](file:///mnt/data/pharmacy-service-v1d.yaml)

Postman Collection: [pharmacy-service.postman\_collection.v2.json](file:///mnt/data/pharmacy-service.postman_collection.v2.json)

Postman Environment: [pharmacy-service.postman\_environment.json](file:///mnt/data/pharmacy-service.postman_environment.json)

TypeScript Client Stubs: [pharmacy-service-client-stubs.ts](file:///mnt/data/pharmacy-service-client-stubs.ts)

# Updated Diagram: Prescription List Flow with Effective Date Policy Filter (Aligned to /v1 & 200/206)

