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

- Feature
- Summary
- Discovery
    - Option 1
    - Option 2
- Solution Sketch
- Orchestration Diagram
- APIs/Swagger
- NFR
- Services
    - Service Logic
    - Service Components
- Client Changes
    - Client side logic
    - Field mapping
    - Any differences between platforms
- Scenarios
- Testing
- Security Review
- Contacts and Impacted Teams/People

## Feature

F215345 -  Correct RX Access for Carved-out Members

## Summary

The team is working to resolve integration and display logic issues related to prescription claims within the CVS Consolidated Claims service, especially in carve-out scenarios where Aetna serves as the pharmacy benefit manager. The challenge centers on ensuring that prescription claims are only shown for coverage periods that align with CVS attestation flags and effective dates, avoiding any display of claims outside valid coverage windows. This requires nuanced front-end logic not documented in API specs, and impacts five key integrations—PLP, PDB, PlaceOrder, Ship Consent, and i90—across 47 APIs. Additionally, the team had to independently develop a crosswalk to bridge gaps in data mapping, and must now coordinate with multiple teams to finalize requirements and prioritize migration efforts, particularly under pressure from urgent timelines and evolving Medicaid carve-out rules in states like North Carolina

## Discovery

### Option 1

Create a centralized service that determines whether prescription claims should be displayed based on a combination of attestation flags, plan effective dates, and carve-out status. This service would act as a gatekeeper, ensuring that only claims within valid coverage periods are shown, regardless of which API is calling for the data. This reduces the need for duplicative logic across front-end applications and ensures consistent behavior across all integrations.

**Technical Specification:**
Develop a middleware service that consumes eligibility data (e.g., attestation flags, plan start dates) and integrates with the CVS Consolidated Claims API. This service would expose a standardized interface to the five key integrations (PLP, PDB, PlaceOrder, Ship Consent, i90), enforcing business rules for claim visibility. It would also maintain a mapping layer (crosswalk) to align internal and external identifiers, and cache logic to optimize performance. This approach minimizes front-end complexity and centralizes compliance logic.

### Option 2

Instead of centralizing the logic, each front-end integration (PLP, PDB, etc.) would implement its own logic to determine whether to show prescription claims. To ensure consistency, a shared validation library would be developed and embedded into each front-end system. This allows teams to move independently but increases the risk of divergence over time.

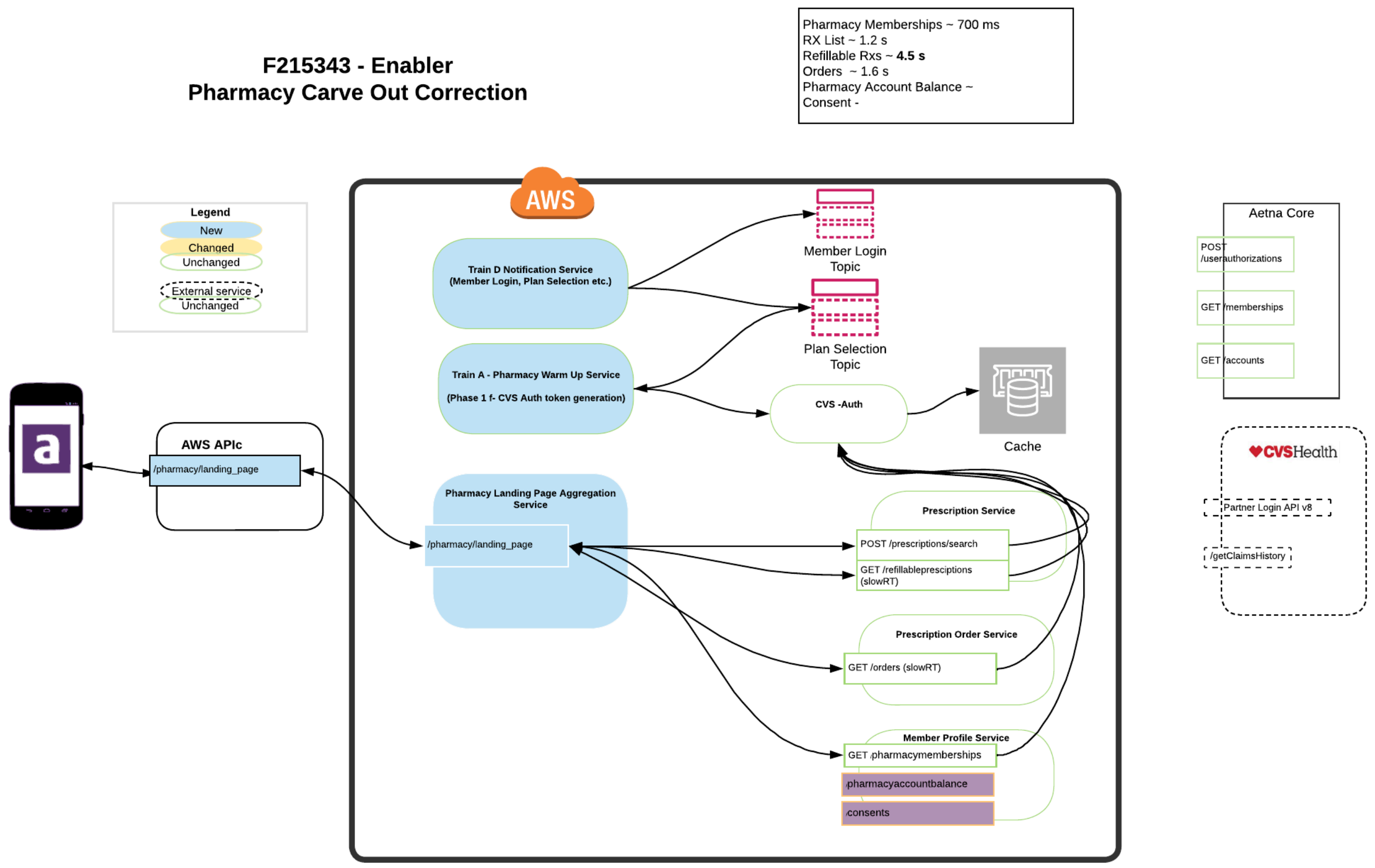
**Technical Specification:**
Build a reusable JavaScript (or language-appropriate) validation module that encapsulates the rules for claim visibility—checking attestation flags, plan dates, and carve-out status. Each front-end would import this module and apply it before rendering prescription data. While this avoids the need for a new service, it requires rigorous version control and testing to ensure all teams stay aligned as rules evolve.

### Selected Option (Option 1)

We chose **Option 1: Centralized Eligibility & Display Logic Service** because it offers a scalable and consistent solution to a complex integration challenge. By centralizing the logic that governs prescription claim visibility—based on attestation flags, plan effective dates, and carve-out status—we eliminate the risk of inconsistent behavior across the five key integrations and 47 APIs. This approach simplifies maintenance, reduces duplication, and ensures that updates to business rules are applied uniformly across all systems.

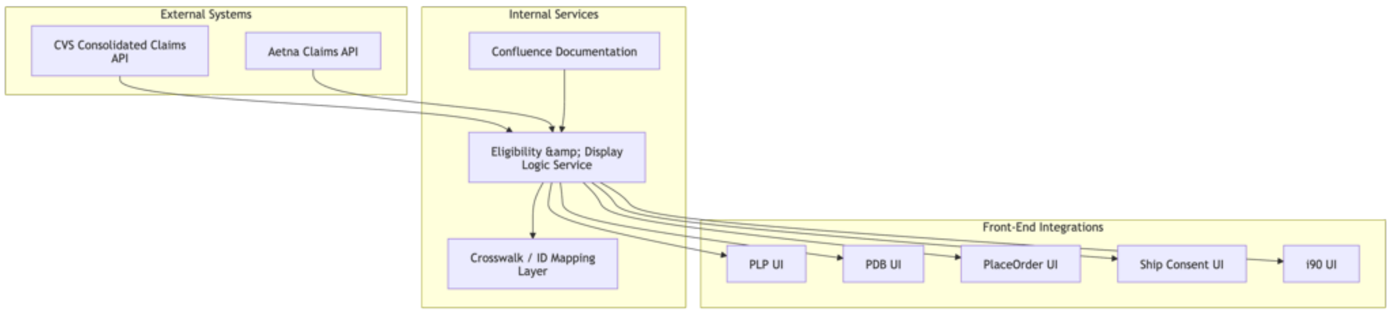
From a technical standpoint, this centralized service acts as a middleware layer that enforces claim display rules before data reaches the front-end. It integrates directly with the CVS Consolidated Claims API and uses a mapping layer to align identifiers across systems. This design minimizes front-end complexity, avoids undocumented logic scattered across interfaces, and supports compliance by ensuring that claims are only shown within valid coverage windows. It also positions us to respond more quickly to future changes in carve-out logic or Medicaid requirements.

## Solution Sketch



## Orchestration Diagram

<<<<<replace with finalized orchestration diagram and sequenc>>>>>



## APIs/Swagger

[https://apiforge.cvshealth.com/apis/editor/Gateway-Specs/agp-api-client-claim-getclaimshistory-v1/1.0.0](https://apiforge.cvshealth.com/apis/editor/Gateway-Specs/agp-api-client-claim-getclaimshistory-v1/1.0.0)

## NFR

**Security & Compliance**

- The service must enforce strict access controls to ensure only authorized systems and users can query or modify claim visibility logic.
- All data exchanges with external APIs (e.g., CVS, Aetna) must be encrypted using TLS, and sensitive data must be masked or tokenized where applicable.

**Performance & Scalability**

- The service must respond to API requests within 200ms under normal load and scale horizontally to support concurrent access from all five integrations and up to 47 APIs.
- Caching mechanisms should be implemented for frequently accessed eligibility data to reduce latency and backend load.

**Maintainability & Extensibility**

- Business rules (e.g., attestation logic, coverage date validation) must be configurable without code changes, ideally via a rules engine or external config.
- The architecture should support easy onboarding of new integrations or changes in carve-out logic without disrupting existing services.

**Observability & Monitoring**

- The service must include logging for all decision points (e.g., why a claim was hidden), and expose metrics for API usage, error rates, and rule evaluations.
- Alerts should be configured for anomalies such as unexpected claim visibility or API failures.

#### Testability & Reliability

- The system must support automated unit, integration, and regression testing for all logic paths, especially around edge cases in coverage and attestation.
- It must maintain 99.9% uptime and include fallback mechanisms if external APIs are unavailable.

## Services

### Service Logic

#### Overview of Flow Through the Service

The centralized service acts as a gatekeeper between front-end integrations and external APIs (CVS, Aetna). When a front-end client requests prescription claim data, the service evaluates eligibility using attestation flags, plan effective dates, and carve-out status. It then determines whether claims should be displayed, ensuring compliance with business rules and preventing exposure of data outside valid coverage windows.

#### Logical Flow Description

1. **Client Request**: A front-end app (e.g., PLP, PDB) sends a request for prescription claims.
2. **Eligibility Evaluation**: The service checks attestation flags, plan dates, and carve-out status.
3. **Mapping & Transformation**: Internal member IDs are mapped to external identifiers using a crosswalk.
4. **API Integration**: The service queries CVS and Aetna APIs for claim data.
5. **Filtering Logic**: Claims are filtered based on coverage dates and attestation status.
6. **Response Formation**: The filtered claims are returned to the client with metadata indicating eligibility status.

#### Transformation / Mapping Table

| Source Attribute | Target Attribute | Notes |
|---|---|---|
| `memberInternalId` | `externalMemberId` | Mapped via crosswalk |
| `planEffectiveDate` | `coverageStartDate` | Used to filter claims |

| | | |
|---|---|---|
| `attestationFlag` | `isCarveOutActive` | Boolean logic for CVS carve-out |
| `claimDate` | `dateOfService` | Used to validate claim visibility |
| `claimType` | `prescriptionClaimType` | Filtered to prescription only |

**Request from Client**

| Attribute | Description |
|---|---|
| `memberInternalId` | Unique ID from front-end |
| `requestType` | e.g., `getPrescriptionClaims` |
| `planEffectiveDate` | Used to validate claim visibility |
| `attestationFlag` | Indicates CVS carve-out status |
| `integrationSource` | e.g., PLP, PDB, PlaceOrder |

**Response to Client**

| Attribute | Description |
|---|---|
| `filteredClaims[]` | Array of eligible prescription claims |
| `eligibilityStatus` | e.g., `eligible`, `ineligible`, `carveOutOnly` |
| `sourceSystem` | CVS or Aetna |
| `claimMetadata` | Includes date of service, pharmacy, status |

## Service Components

### ODM (Core Rules Engine)

- **Purpose**: Evaluate prescription claim visibility based on attestation flags, plan effective dates, and carve-out status.
- **Rules Summary**:
  - Inputs: `attestationFlag`, `planEffectiveDate`, `claimDate`, `carveOutStatus`
  - Outputs: `claimVisibilityStatus` (`eligible`, `ineligible`, `carveOutOnly`)
- **Implementation**: Rules defined in a configurable rules engine (ODM or equivalent), with externalized logic for easy updates.

### Extreme Scale (Caching in Core Services)

- **Purpose**: Improve performance and reduce latency for frequent eligibility checks.
- **Cache Key**: `memberInternalId + planEffectiveDate + attestationFlag`
- **Expiry**: 15 minutes for dynamic data; 24 hours for static mappings
- **Grid Size**: Scalable to support 10M+ member records

### ACE/EDB Services

- **Purpose**: Support eligibility lookups and member data enrichment
- **Integration**: Service calls to ACE/EDB for member metadata and coverage validation
- **Deployment**: Cloud-hosted (AWS)

### Cloud Infrastructure

- **Platform**: AWS
- **Services Used**:
  - AWS Lambda (business logic execution)
  - Amazon API Gateway (service exposure)
  - Amazon DynamoDB (crosswalk and metadata storage)
  - Amazon ElastiCache (caching layer)
  - Amazon S3 (logging and audit trail)

### AEI Teams Supporting the Service

- **Eligibility Logic & Rules**: AEI Eligibility Team
- **API Gateway & Integration**: AEI Integration Services Team

- **Infrastructure & DevOps**: AEI Cloud Platform Team

#### CMS (Contentful or Other)

- **Content Model**: Not applicable for this service (no user-facing content)

#### Database

- **Technology**: Amazon DynamoDB
- **Schema/Data Stored**:
  - Member ID mappings
  - Attestation flags
  - Coverage dates
  - Claim metadata
- **Interaction**: Queried by the logic service during claim evaluation
- **Anticipated Size**: ~50M records across all integrations

#### EDA / ETL Processes

- **Purpose**: Sync attestation flags and coverage data from external systems (CVS, Aetna)
- **Frequency**: Daily batch updates + real-time event triggers
- **Tools**: AWS Glue, EventBridge

## Client Changes

### Client-Side Logic Summary

Front-end applications (PLP, PDB, PlaceOrder, Ship Consent, i90) must integrate with the centralized eligibility service to determine whether prescription claims should be displayed. The client sends a structured request containing member and plan data, and receives a filtered response indicating claim visibility. All logic for eligibility evaluation is handled server-side, minimizing complexity on the client.

#### 🔗 Field Mapping Table

| UI Field | Request Attribute | Notes |
|---|---|---|
| Member ID | `memberInternalId` | Passed from authenticated session |
| Plan Start Date | `planEffectiveDate` | Used to validate claim visibility |

| | | |
|---|---|---|
| Carve-Out Status (CVS flag) | `attestationFlag` | Boolean flag from member profile |
| Integration Source | `integrationSource` | e.g., PLP, PDB, PlaceOrder |
| Claim Type | `claimType` | Should default to `prescription` |

**Response Mapping Table**

| Response Attribute | UI Display Field | Notes |
|---|---|---|
| `filteredClaims[]` | Prescription Claims List | Only eligible claims shown |
| `eligibilityStatus` | Claim Visibility Indicator | e.g., `eligible`, `carveOutOnly`, `ineligible` |
| `sourceSystem` | Data Source Label | CVS or Aetna |
| `claimMetadata` | Claim Details | Includes date of service, pharmacy, status |

**Platform Differences**

| Platform | Integration Notes |
|---|---|
| Web | Full integration with centralized service; caching recommended for session performance |
| iOS | Use native networking layer; ensure secure token handling and error fallback |
| Android | Similar to iOS; use Retrofit or equivalent for API calls; validate |

| | |
|---|---|
| | date formatting |

#### Supporting Materials

- **Screenshots**: [Insert screenshots of UI fields and claim display logic]
- **Miro Link**: [Insert link to Miro board showing flow and UI mapping]
- **API Spec Reference**: [Link to OpenAPI/Swagger spec for eligibility service]

## Scenarios

*Identify use cases that are critical for this feature in mapping, logic, testing.*

## Testing

*Please explain how this solution/update should be tested.  Any aspects of the solution specific to testability.*

## Security Review

*Provide any notes, changes done as part of security review.  Security reviews should be done especially when something non-standard is happening.  Third party integration, some new technology/component, SDK recommendation, SFTP, PII/PHI storage, etc ... please do a security review with members of security team and provide date, findings, changes here.*

## Contacts and Impacted Teams/People

#### Implementation & Architecture

- **Dworkin, Amram** – Lead Architect (Primary owner of service design and integration)
- **AEI Integration Services Team** – API Gateway, service orchestration, and external API connectivity
- **AEI Eligibility Team** – Rules engine configuration and eligibility logic
- **AEI Cloud Platform Team** – AWS infrastructure, caching, and deployment support

#### Testing & QA

- **Front-End QA Team** – Responsible for validating claim visibility logic across PLP, PDB, PlaceOrder, Ship Consent, and i90
- **API QA Team** – Ensures consistent behavior across service endpoints and integrations

#### Front-End Development

- **PLP Development Team**
- **PDB Development Team**
- **PlaceOrder Development Team**
- **Ship Consent Development Team**
- **i90 Development Team**

#### Data & ETL

- **Data Engineering Team** – Supports crosswalk mapping, attestation flag sync, and coverage data ingestion
- **ETL Operations Team** – Manages batch updates and real-time triggers from CVS and Aetna

#### Business & Product Stakeholders

- **Harrington, Joan K** – Manager, prioritization and stakeholder alignment
- **Behrmann, Ian J**, **Almustafa, Ali**, **Tang, Jennifer J**, **Kieterling, Kamil** – Top collaborators involved in requirements gathering and validation