Here are some high-level solutions architect questions about this solution, focusing on aspects not explicitly covered or needing further detail in the provided document:

**Overall Architecture & Strategy:**

1. **Future State Vision:** This document focuses on "routing plan document list and PDF retrieval through AWS services." What is the broader, long-term vision for plan document management and retrieval, and how does this feature align with that vision? Are there other current or future capabilities (e.g., proactive document delivery, enhanced personalization) that this architecture needs to consider?
2. **AWS Services Evolution:** The document mentions using S3. Are there other AWS services being considered for future enhancements or to optimize existing functionalities (e.g., using Lambda for serverless processing, DynamoDB for metadata, ECS/EKS for containerized services, AWS Glue for data transformation, etc.)?
3. **Monolith to Microservices:** Is this initiative part of a larger strategy to decompose existing monolithic applications into microservices, or is it a standalone optimization? How will this architecture integrate with or influence other existing or planned microservices?
4. **Cost Optimization:** Beyond the caching benefits, what specific strategies are in place to monitor and optimize AWS costs associated with S3 storage, data transfer, API Gateway usage, and Lambda invocations, especially considering potential high volume?
5. **Multi-Region / Disaster Recovery:** The S3 buckets are specified in us-east-1. What is the disaster recovery (DR) strategy for this solution? Will it be deployed in multiple AWS regions, and if so, how will data synchronization and failover be handled?

**Data & Caching:**

1. **Cache Invalidation Strategy (Beyond TTL):** The document mentions an hourly TTL for the plan document list cache. What is the strategy for actively invalidating cached data if underlying plan document information changes more frequently than the TTL, especially for critical updates or corrections?
2. **Data Consistency for PDF Retrieval:** When a PDF is not found in S3 and retrieved from the core endpoint, it's then stored in S3. What mechanisms are in place to ensure data consistency between the core endpoint and the S3 bucket, particularly if the core data changes *after* a PDF has been cached in S3 but *before* its TTL expires?
3. **Data Governance & Compliance:** What are the data governance policies for the plan documents stored in S3, including data retention, encryption at rest and in transit, and access control? How does this align with industry regulations (e.g., HIPAA, if applicable) and internal compliance requirements?
4. **Cache Key Complexity & Evolution:** The document highlights the complexity of Medicare cache keys. How will the evolution of Medicare Group ANOC solutions and the potential introduction of a "Master Plan ID" impact the caching strategy and the definition of cache keys in the future? What is the plan for backward compatibility during such transitions?

**Security & Operations:**

1. **Authentication and Authorization:** The document mentions "App Token Core Endpoints." How is authentication and authorization handled for accessing the AWS APIc endpoints and the underlying S3 buckets? What security best practices (e.g., IAM roles, least privilege, API Gateway authorization) are being implemented?
2. **Monitoring and Alerting:** What comprehensive monitoring and alerting mechanisms will be in place to track the health, performance, and security of the AWS services involved (API Gateway, Lambda, S3)? What key metrics will be monitored, and what are the thresholds for alerts?
3. **Logging and Auditing:** What logging solutions are being used (e.g., CloudWatch Logs, S3 access logs), and how will these logs be analyzed for operational insights, troubleshooting, and security auditing?
4. **Deployment and Release Management:** How will deployments and rollbacks be managed for the AWS components of this solution? What CI/CD pipelines are being utilized, and what are the testing strategies (unit, integration, performance, security)?
5. **Error Handling and Resilience:** The document mentions handling empty files and HHL not signed scenarios. What is the comprehensive error handling strategy for all potential failures within the AWS services and interactions with core endpoints? How does the solution ensure resilience (e.g., retries, dead-letter queues, circuit breakers)?

**Performance & Scalability:**

1. **Performance Benchmarking:** Beyond the stated rate limits, what are the target latency and throughput metrics for the plan document list and PDF retrieval endpoints? What performance benchmarking will be conducted, and how will the solution scale to handle peak loads?
2. **Throttling and Rate Limiting (AWS Side):** Are there any AWS-level throttling or rate-limiting configurations being applied to protect the solution from abusive traffic or to manage costs, beyond the existing core API rate limits?
3. **Concurrency Management:** How is concurrency managed within the Lambda functions or other compute services to prevent resource exhaustion or performance degradation under high load?

**Development & Integration:**

1. **API Versioning:** The API endpoints mention /v4 and /v5. What is the API versioning strategy, and how will backward compatibility be maintained for existing consumers when new versions are introduced?
2. **Inter-Service Communication:** How will different services involved in this architecture (e.g., AH Client BFF, benefits domain service, plan doc search, Core-Proxy) communicate with each other? What protocols and patterns are being used (e.g., REST, event-driven, message queues)?
3. **Dependency Management:** How are dependencies on core APIs and other internal services managed? What strategies are in place for handling outages or performance degradation of these dependencies?

**Checklist for Creating a New BFF Endpoint:**

1. **Feature Grooming & Requirements Definition:**
   * Product and Design teams develop comprehensive feature requirements and user experience mocks.
   * Socialize requirements and mocks with the broader team for feedback.
   * Backend Architecture (BE Arch) determines if a new API or a substantial change to an existing API is needed to support the feature.
2. **Backend Architecture (BE Arch) Enabler:**
   * If a new or substantially changed API is required, BE Arch creates an Arch Enabler.
   * BE Arch designs the API (or changes an existing one) and reviews the API design and contract with Client Architecture (Client Arch).
3. **Data Mapping & API Responsibility Definition:**
   * Client Arch, Design, and Product collaborate to map available data from BE services to the mocks/user experience.
   * Technical teams (BE Arch, Client Arch, Engineering) decide on the specific responsibilities of the new BFF API versus the existing Backend Services. This is crucial for defining the BFF's scope (e.g., orchestration, data transformation for client needs).
4. **Client Architecture (Client Arch) Enabler:**
   * Client Arch begins designing the **BFF API** based on the agreed-upon responsibilities. This includes defining the client-facing API contract.
   * The design should focus on client-centric logic and adaptability across platforms.
   * Engineering teams should be consulted throughout this design phase.
5. **BFF API Contract Development (First Iteration):**
   * Client Arch (with Engineering consultation) develops the initial version of the BFF API contract. This is a new process step.
6. **Feature Implementation (Engineering Phase):**
   * Engineering teams develop the BFF API based on the designed contract.
   * They implement the client-side features, consuming the new BFF API.
   * Leverage prior early involvement to minimize churn in this phase.
7. **Ongoing BFF API Contract Maintenance:**
   * Engineering becomes accountable for any subsequent updates to the BFF API contracts after the initial development by Client Arch.
   * Engineering should consult and review with Client Arch as needed for these updates.

**Key Principles to Follow:**

* **Contract-First Strategy:** Design the API contract before extensive implementation.
* **Client Ownership:** The client team (specifically Client Arch and eventually Engineering) takes primary ownership of the BFF API.
* **Early Cohesion & End-to-End Architecture:** Foster collaboration among all teams (Product, Design, BE Arch, Client Arch, Engineering) from the initial grooming phase.
* **Best Effort Warming:** If applicable (as per the PDF download use case), implement best-effort warming strategies for caches.

This checklist aims to provide a structured approach to integrating new BFF endpoints within the existing service landscape at Aetna.

Generate Audio Overview

Video Deep Research Canvas

Gemini can make mistakes, so double-check it

Google Account

Amram Dworkin

[amram.dworkin@gmail.com](mailto:amram.dworkin@gmail.com)

Use microphone

Add files

Generate with Veo 3

Get in-depth answers

Files in this chat

Copy response

More

Expand text

Edit text

Expand menu

New chat

Explore Gems

AWS Solution Architect Questions

Gitignore .env Except .env-clean

Cross-Platform Build Scripting Solutions

Testing Gmail Authorization Functionality

Fix TypeScript Google Config Error

Settings & help

Create docs and apps