# OpenPlatform Plan Documents BFF Service

## Overview

The **OpenPlatform Plan Documents BFF (Backend for Frontend)** service provides a streamlined interface for retrieving plan documents and their associated metadata within the Aetna OpenPlatform ecosystem. This service acts as an intermediary layer that simplifies access to plan documents by abstracting the complexity of underlying backend systems.

### Service Details

* **Title**: OpenPlatform - Plan Documents BFF
* **Version**: 1.0.4
* **Description**: Provides plan document list and PDF retrieval for the OpenPlatform
* **Base Path**: /sa/plandocs/v1

## Authentication & Security

The service implements a dual authentication mechanism to ensure secure access to plan documents:

### 1. Bearer Token Authentication

* **Type**: HTTP Bearer authentication
* **Format**: JWT (JSON Web Token)
* **Header**: Authorization: Bearer <jwt\_token>
* **Purpose**: Primary authentication mechanism for API access

### 2. ID Token Header Authentication

* **Type**: Custom header parameter
* **Header Name**: id\_token
* **Format**: JWT containing user context and authorization information
* **Required**: Yes
* **Purpose**: Provides detailed user context and authorization information

#### ID Token Payload Structure

The id\_token contains comprehensive user information including:

{  
 "ae\_version": "1.1.0",  
 "iss": "https://openid.aetna.com/consumer",  
 "sub": "031RCS6NTYHFKX25DE89@aetnae.com",  
 "aud": "89d435af-00b0-4cb6-9a35-f3189f5adc55",  
 "exp": 1753561541,  
 "iat": 1753557941,  
 "given\_name": "BAILEY",  
 "family\_name": "SCHEUERMAN",  
 "acr": "http://consumer.aetna.com/assurance/loa-2",  
 "ae\_dgn": "CN=DMT-S-W265416171,OU=Members,OU=External,DC=aetheq,DC=aetnaeq,DC=com",  
 "ae\_hcr": "nextGenMember",  
 "ae\_accountId": "1~DMT-S-W265416171",  
 "ae\_busIndID": [  
 "globalIdentifier",  
 "60005~6803568937376433212",  
 "preferredProxyId",  
 "15~QS3YXBBBHPXZ"  
 ],  
 "ae\_impAUD": "",  
 "ae\_impHCR": "",  
 "ae\_impACR": "",  
 "ae\_impDGN": "",  
 "ae\_impAccountId": "",  
 "ae\_impBusIndID": [],  
 "ae\_impGrantedLOA": ""  
}

**Key Token Fields**:

* ae\_version: Aetna Enterprise version
* sub: Subject identifier (user ID)
* ae\_hcr: Health Care Role (e.g., “nextGenMember”)
* ae\_accountId: Account identifier
* ae\_busIndID: Business individual identifiers
* exp: Token expiration time
* iat: Token issued at time

## API Endpoints

### 1. Plan Document List Retrieval

**Endpoint**: POST /sa/plandocs/v1/list

**Purpose**: Retrieves a comprehensive list of available plan documents for a specific user and membership.

#### Authentication Requirements

* **Bearer Token**: Required in Authorization header
* **ID Token**: Required in id\_token header

#### Request Headers

Authorization: Bearer <jwt\_token>  
id\_token: <jwt\_token\_with\_user\_context>  
Content-Type: application/json

#### Response Structure

**Success Response (200)**:

{  
 "data": [  
 {  
 "planId": "751133+BA+2",  
 "documentName": "Plan Document (Spanish)",  
 "documentId": "70023~13910995",  
 "documentUrl": "/sa/plandocs/v1/retrieve/70023~13910995"  
 },  
 {  
 "planId": "751133+BA+2",  
 "documentName": "Summary of Benefits and Coverage",  
 "documentId": "70023~13910996",  
 "documentUrl": "/sa/plandocs/v1/retrieve/70023~13910996"  
 }  
 ],  
 "readCommunicationsResponse": "HHL Not signed"  
}

**Response Fields**:

* data: Array of simplified document objects
* readCommunicationsResponse: Special messaging for the response

**Document Object Fields**:

* planId: Plan identifier
* documentName: Name of the document
* documentId: Unique document identifier
* documentUrl: Internal URL to retrieve the PDF via BFF (includes documentId)

**readCommunicationsResponse Valid Values**:

* null: Request succeeded (normal case)
* "No docs available": No PlanDocsMsgAvail but no docs returned
* "Your employer can provide these documents to you": Special messaging for employer-provided documents
* "HHL Not signed": Special messaging for HHL not signed scenario

#### Error Responses

**Bad Request (400)**:

{  
 "httpCode": "400",  
 "httpMessage": "Bad Request",  
 "moreInformation": "Invalid request parameters."  
}

**Unauthorized (401)**:

{  
 "httpCode": "401",  
 "httpMessage": "Unauthorized",  
 "moreInformation": "Invalid or missing authentication."  
}

**Forbidden (403)**:

{  
 "httpCode": "403",  
 "httpMessage": "Forbidden",  
 "moreInformation": "Insufficient permissions to access plan documents."  
}

**Internal Server Error (500)**:

{  
 "httpCode": "500",  
 "httpMessage": "Internal Server Error",  
 "moreInformation": "An unexpected error occurred while processing the request."  
}

### 2. Plan Document PDF Retrieval

**Endpoint**: POST /sa/plandocs/v1/retrieve

**Purpose**: Retrieves a specific plan document PDF file based on the document identifier.

#### Request Structure

{  
 "documentId": "70023~13910995"  
}

**Required Fields**:

* documentId: Corresponds to communicationContentResourceId in the backend (communicationContentIdentifier.resourceId)

#### Response Structure

**Success Response (200)**:

* **Content-Type**: application/pdf
* **Body**: Binary PDF file content

#### Error Responses

**Bad Request (400)**:

{  
 "httpCode": "400",  
 "httpMessage": "Bad Request",  
 "moreInformation": "Invalid documentId provided."  
}

**Internal Server Error (500)**:

{  
 "httpCode": "500",  
 "httpMessage": "Internal Server Error",  
 "moreInformation": "Failed to retrieve the requested document."  
}

## Data Models

### PlanDocListResponse

Simplified response containing an array of document objects and optional special messaging.

### PlanDocRetrieveRequest

Simple request containing the document identifier for PDF retrieval.

### Error

Standard error response structure with HTTP code, message, and additional information.

### IdTokenPayload

Comprehensive JWT payload structure containing user context, authentication information, and authorization details.

## Use Cases

### 1. Member Portal Integration

Members can view available plan documents in their portal, with the ability to download PDFs directly. The simplified response structure makes it easy to display document lists.

### 2. Mobile Application Support

Mobile apps can display document lists and provide PDF viewing capabilities with minimal data transfer.

### 3. Customer Service Tools

Customer service representatives can access and share plan documents with members using the dual authentication system.

### 4. Compliance and Regulatory Requirements

Ensures proper access to required plan documents for regulatory compliance with comprehensive audit trails.

## Integration Patterns

### Frontend Integration with Authentication

// Example: Fetching plan document list with dual authentication  
const response = await fetch('/sa/plandocs/v1/list', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + bearerToken,  
 'id\_token': idToken  
 }  
});  
  
const documents = await response.json();  
  
// Handle special messaging  
if (documents.readCommunicationsResponse) {  
 console.log('Special message:', documents.readCommunicationsResponse);  
}  
  
// Display documents  
documents.data.forEach(doc => {  
 console.log(`${doc.documentName} (${doc.documentId})`);  
});

### PDF Download

// Example: Downloading a specific document  
const response = await fetch('/sa/plandocs/v1/retrieve', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + bearerToken  
 },  
 body: JSON.stringify({  
 documentId: '70023~13910995'  
 })  
});  
  
if (response.ok) {  
 const pdfBlob = await response.blob();  
 const url = URL.createObjectURL(pdfBlob);  
 window.open(url);  
} else {  
 const error = await response.json();  
 console.error('Download failed:', error.moreInformation);  
}

### Error Handling

// Comprehensive error handling  
async function fetchPlanDocuments() {  
 try {  
 const response = await fetch('/sa/plandocs/v1/list', {  
 method: 'POST',  
 headers: {  
 'Content-Type': 'application/json',  
 'Authorization': 'Bearer ' + bearerToken,  
 'id\_token': idToken  
 }  
 });  
  
 if (response.status === 401) {  
 // Handle authentication failure  
 console.error('Authentication failed - please log in again');  
 return;  
 }  
  
 if (response.status === 403) {  
 // Handle authorization failure  
 console.error('Insufficient permissions to access plan documents');  
 return;  
 }  
  
 if (!response.ok) {  
 const error = await response.json();  
 console.error('Request failed:', error.moreInformation);  
 return;  
 }  
  
 const documents = await response.json();  
 return documents;  
 } catch (error) {  
 console.error('Network error:', error);  
 }  
}

## Security Considerations

### Authentication Layers

* **Bearer Token**: Primary authentication for API access
* **ID Token**: Provides detailed user context and authorization information
* **Token Validation**: Both tokens are validated on every request

### Authorization

* Document access is restricted based on user membership and roles
* Health Care Role (ae\_hcr) determines access permissions
* Account ID (ae\_accountId) ensures proper data isolation

### Data Protection

* Sensitive plan information is protected through proper authorization
* Audit trails are maintained for document access
* Token expiration ensures secure session management

## Performance Considerations

### Simplified Response Structure

* Reduced payload size compared to previous versions
* Faster parsing and rendering in frontend applications
* Optimized for mobile and web applications

### Caching Strategy

* Document lists can be cached based on user context
* PDF retrieval is optimized for large file handling
* Token caching reduces authentication overhead

## Error Handling Strategy

The service provides comprehensive error handling with specific error codes:

* **400 Bad Request**: Invalid input parameters or missing required fields
* **401 Unauthorized**: Invalid or missing authentication tokens
* **403 Forbidden**: Insufficient permissions to access requested resources
* **500 Internal Server Error**: Unexpected server-side errors

## Future Enhancements

* Support for document search and filtering
* Pagination for large document lists
* Document preview capabilities
* Multi-language document support
* Document versioning and history tracking
* Enhanced caching strategies
* Real-time document availability updates