API Virtualization Service (AVS) - Enterprise Integration Platform [Draft]

Overview

The API Virtualization Service provides a robust platform for creating, managing, and deploying virtual APIs across the enterprise. This service enables teams to develop and test applications without dependencies on production endpoints.

Key Features

- Dynamic API virtualization
- Scenario-based response management
- Real-time response customization
- Enterprise-grade security
- Comprehensive monitoring
- Self-service capabilities

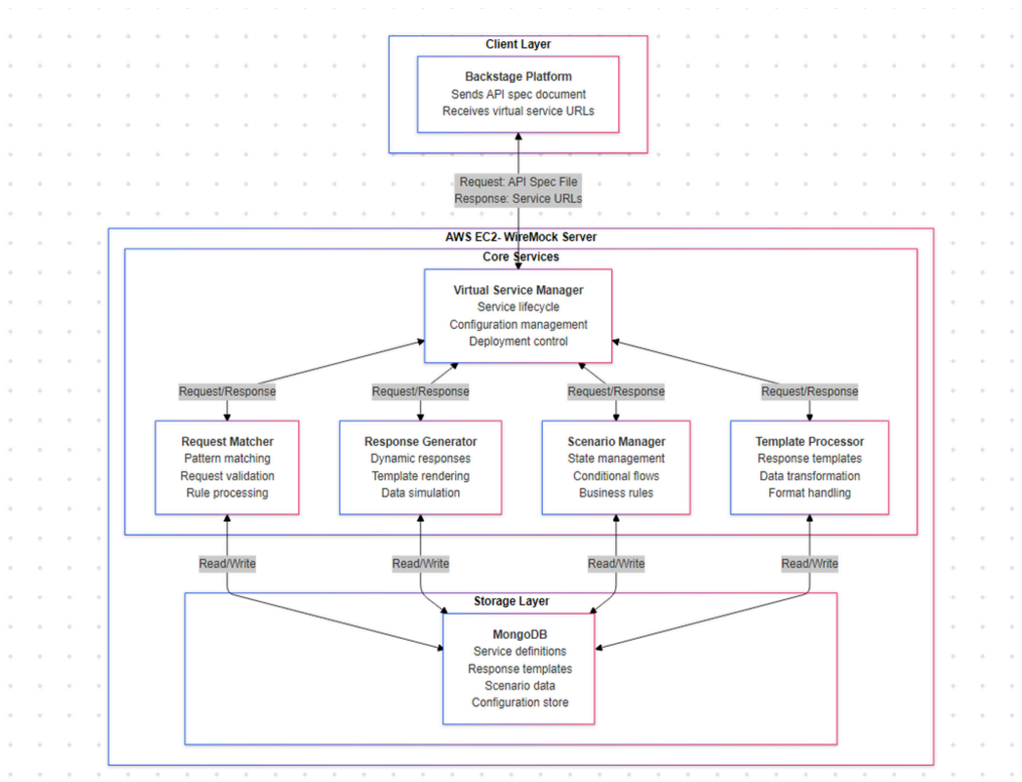
How It Works

1. **Service Creation**
    - Teams submit virtual service requests via REST APIs
    - System validates and provisions the virtual endpoint
    - Automatic deployment to the virtualization platform
2. **Response Management**
    - Template-based response configuration
    - Dynamic data generation
    - Conditional response patterns
    - State-based scenarios
3. **Integration Flow**
    - Client applications connect to virtual endpoints
    - Request matching based on configured patterns
    - Response generation using templates
    - Monitoring and logging of interactions

Use Cases

- Application development and testing
- Performance testing
- Integration testing
- Training environments
- Demo environments

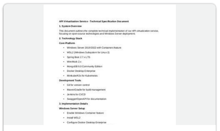
Architecture Components



Integration Guidelines

- REST API specifications
- Authentication requirements
- Request/Response formats
- Error handling patterns
- Best practices

API Virtualization Service - Technical Specification



API Virtualiz... ce.docx
21 Nov 2024, 05:21 PM

List of essential information required in the API specification document for creating virtual services:

Required Information (Must Have)

1. Base API Information
   - API Name/Title
   - Version
   - Base URL structure
2. Endpoint Specifications
   - Complete endpoint paths

    - HTTP methods
    - Path parameters
3. Request Details
    - Request body schema
    - Required fields
    - Field data types
    - Sample request payloads
4. Response Details
    - Response status codes
    - Response body schema
    - Sample response payloads
5. Error Scenarios
    - Error codes
    - Sample error responses

Optional Information (Nice to Have)

1. Authentication Details (if public API)
2. Business Rules (for simple APIs)
3. Test Scenarios (can be developed iteratively)
4. Performance Expectations
5. Advanced Data Requirements

[Draft] Request and Response Payload structure -

1. **Create Virtual Service API**

```
// POST /api/v1/virtualservice (multipart/form-data)

// Form Fields:
{
    "projectName": "payment-gateway",
    "projectId": "PG-2024",
    "teamName": "payments-team",
    "environment": "dev",
    "description": "Payment Processing API Virtual Service",
    "owner": "john.doe@questdiagnostics.com",
    "tags": ["payments", "gateway", "api"],
    "expiryDate": "2024-12-31"
}

// File Attachment:
apiSpec: api-specification.yaml/json (OpenAPI/Swagger specification file)

// Response:
{
    "serviceId": "vs-123456",
    "status": "CREATED",
    "virtualServiceUrl": "http://virtual-service/payment-gateway",
    "created": "2024-01-20T10:30:00Z",
    "expiresOn": "2024-12-31T23:59:59Z",
    "metadata": {
        "projectId": "PG-2024",
        "environment": "dev",
        "endpoints": [
            "/api/v1/payments",
            "/api/v1/refunds"
        ]
    }
}

```

2. **Update Virtual Service API**

```
// PUT /api/v1/virtualservice/{serviceId}
Request:
{
    "responseTemplate": {
        "status": 200,
        "body": {
            "transactionId": "UUID",
            "status": "SUCCESS",
            "timestamp": "datetime",
            "additionalInfo": "string"
        }
    }
}

Response:
{
    "serviceId": "vs-123456",
    "status": "UPDATED",
    "timestamp": "2024-01-20T10:35:00Z"
}

```

3. **Add New Scenario API**

```
// POST /api/v1/virtualservice/{serviceId}/scenarios
Request:
{
    "scenarioName": "timeout_error",
    "responseTemplate": {
        "status": 504,
        "body": {
            "error": "GATEWAY_TIMEOUT",
            "message": "Service unavailable"
        }
    },
    "delay": 5000
}

Response:
{
    "serviceId": "vs-123456",
    "scenarioId": "scn-789",
    "status": "CREATED",
    "timestamp": "2024-01-20T10:40:00Z"
}

```

4. **Get Service Status API**

```
// GET /api/v1/virtualservice/{serviceId}
Response:
{
    "serviceId": "vs-123456",
    "status": "ACTIVE",
    "statistics": {
        "totalRequests": 1000,
        "successCount": 850,
        "errorCount": 150,
        "averageResponseTime": 120
    },
    "scenarios": [
        {
            "name": "success_case",
            "hitCount": 850
        },
        {
            "name": "insufficient_funds",
            "hitCount": 150
        }
    ],
    "lastUpdated": "2024-01-20T10:45:00Z"
}

```