Singapore G3 Payment Platform - Service Design Documentation

Table of Contents

1. 1. Fast Ddi Validation Service
2. 2. Fast Inwd Processor Service
3. 3. Fast Accountlookup Service
4. 4. Fast Referencedata Service

Fast DDI Validation Service - Design Document

Overview

**The Fast DDI Validation Service is a gRPC-based microservice responsible for validating enriched PACS messages and publishing them to Kafka for downstream orchestration. It serves as the bridge between synchronous gRPC processing and asynchronous Kafka-based orchestration in the Singapore G3 Payment Platform.**

Key Responsibilities

* Validate enriched payment messages for Singapore market compliance
* Perform XSD schema validation for PACS messages
* Currency validation (SGD-specific rules)
* Country validation (Singapore market compliance)
* Convert XML to JSON format for downstream processing
* Publish validated messages to Kafka for orchestration

Service Details

* \*\*Service Type\*\*: gRPC Service
* \*\*Port\*\*: 50053
* \*\*Package\*\*: `gpp.g3.ddivalidation`
* \*\*Technology Stack\*\*: TypeScript, gRPC, Kafka
* \*\*Environment\*\*: Singapore G3 Payment Platform

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sequence Diagram

<?xml version="1.0" encoding="UTF-8"?>

<mxfile host="app.diagrams.net" modified="2024-01-01T00:00:00.000Z" agent="5.0" version="24.7.17" etag="draw.io-diagram" type="device">

<diagram name="DDI Validation Sequence" id="ddi-validation-sequence">

<mxGraphModel dx="1422" dy="794" grid="1" gridSize="10" guides="1" tooltips="1" connect="1" arrows="1" fold="1" page="1" pageScale="1" pageWidth="1169" pageHeight="827" math="0" shadow="0">

<root>

<mxCell id="0" />

<mxCell id="1" parent="0" />

<!-- Participants -->

<mxCell id="client" value="fast-inwd-processor-service" style="shape=umlLifeline;perimeter=lifelinePerimeter;whiteSpace=wrap;html=1;container=1;collapsible=0;recursiveResize=0;outlineConnect=0;" vertex="1" parent="1">

<mxGeometry x="80" y="80" width="180" height="500" as="geometry" />

</mxCell>

<mxCell id="ddi" value="fast-ddi-validation-service" style="shape=umlLifeline;perimeter=lifelinePerimeter;whiteSpace=wrap;html=1;container=1;collapsible=0;recursiveResize=0;outlineConnect=0;" vertex="1" parent="1">

<mxGeometry x="320" y="80" width="180" height="500" as="geometry" />

</mxCell>

<mxCell id="kafka" value="Kafka Broker" style="shape=umlLifeline;perimeter=lifelinePerimeter;whiteSpace=wrap;html=1;container=1;collapsible=0;recursiveResize=0;outlineConnect=0;" vertex="1" parent="1">

<mxGeometry x="560" y="80" width="120" height="500" as="geometry" />

</mxCell>

<mxCell id="orch" value="fast-orchestrator-service" style="shape=umlLifeline;perimeter=lifelinePerimeter;whiteSpace=wrap;html=1;container=1;collapsible=0;recursiveResize=0;outlineConnect=0;" vertex="1" parent="1">

<mxGeometry x="720" y="80" width="160" height="500" as="geometry" />

</mxCell>

<!-- Messages -->

<mxCell id="msg1" value="ValidateEnrichedMessage(request)" style="html=1;verticalAlign=bottom;startArrow=oval;endArrow=block;startSize=8;curved=0;rounded=0;" edge="1" parent="1">

<mxGeometry width="80" relative="1" as="geometry">

<mxPoint x="170" y="160" as="sourcePoint" />

<mxPoint x="410" y="160" as="targetPoint" />

</mxGeometry>

</mxCell>

<mxCell id="note1" value="Extract message data" style="shape=note;whiteSpace=wrap;html=1;backgroundOutline=1;fontColor=#000000;darkOpacity=0.05;fillColor=#FFF9B2;strokeColor=none;fillStyle=solid;direction=west;gradientDirection=north;gradientColor=#FFF2A1;shadow=1;size=20;" vertex="1" parent="1">

<mxGeometry x="440" y="180" width="120" height="40" as="geometry" />

</mxCell>

<!-- Self-calls for validation steps -->

<mxCell id="parse" value="Parse XML payload" style="html=1;verticalAlign=bottom;startArrow=oval;endArrow=block;startSize=8;curved=0;rounded=0;" edge="1" parent="1">

<mxGeometry width="80" relative="1" as="geometry">

<mxPoint x="410" y="240" as="sourcePoint" />

<mxPoint x="450" y="240" as="targetPoint" />

</mxGeometry>

</mxCell>

<!-- Success and error flows -->

<mxCell id="publishKafka" value="Publish to validated-messages topic" style="html=1;verticalAlign=bottom;startArrow=oval;endArrow=block;startSize=8;curved=0;rounded=0;" edge="1" parent="1">

<mxGeometry width="80" relative="1" as="geometry">

<mxPoint x="410" y="400" as="sourcePoint" />

<mxPoint x="620" y="400" as="targetPoint" />

</mxGeometry>

</mxCell>

<mxCell id="responseSuccess" value="ValidationResponse(success=true)" style="html=1;verticalAlign=bottom;startArrow=oval;endArrow=block;startSize=8;curved=0;rounded=0;" edge="1" parent="1">

<mxGeometry width="80" relative="1" as="geometry">

<mxPoint x="410" y="430" as="sourcePoint" />

<mxPoint x="170" y="430" as="targetPoint" />

</mxGeometry>

</mxCell>

</root>

</mxGraphModel>

</diagram>

</mxfile>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class Diagram

<?xml version="1.0" encoding="UTF-8"?>

<mxfile host="app.diagrams.net" modified="2024-01-01T00:00:00.000Z" agent="5.0" version="24.7.17" etag="draw.io-diagram" type="device">

<diagram name="DDI Validation Classes" id="ddi-validation-classes">

<mxGraphModel dx="1422" dy="794" grid="1" gridSize="10" guides="1" tooltips="1" connect="1" arrows="1" fold="1" page="1" pageScale="1" pageWidth="1169" pageHeight="827" math="0" shadow="0">

<root>

<mxCell id="0" />

<mxCell id="1" parent="0" />

<!-- DDIValidationService -->

<mxCell id="ddiService" value="DDIValidationService" style="swimlane;fontStyle=1;align=center;verticalAlign=top;childLayout=stackLayout;horizontal=1;startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;resizeLast=0;collapsible=1;marginBottom=0;" vertex="1" parent="1">

<mxGeometry x="200" y="100" width="320" height="280" as="geometry" />

</mxCell>

<mxCell id="ddiFields" value="+expectedCurrency: string&#xa;+expectedCountry: string&#xa;+isTestMode: boolean" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="ddiService">

<mxGeometry y="26" width="320" height="54" as="geometry" />

</mxCell>

<mxCell id="ddiLine" value="" style="line;strokeWidth=1;fillColor=none;align=left;verticalAlign=middle;spacingTop=-1;spacingLeft=3;spacingRight=3;rotatable=0;labelPosition=right;points=[];portConstraint=eastwest;strokeColor=inherit;" vertex="1" parent="ddiService">

<mxGeometry y="80" width="320" height="8" as="geometry" />

</mxCell>

<mxCell id="ddiMethods" value="+validateEnrichedMessage(request): Promise&lt;ValidationResponse&gt;&#xa;+performValidations(parsedXml, enrichmentData): Promise&lt;ValidationResult&gt;&#xa;+validateCurrency(currency): CurrencyValidation&#xa;+validateCountry(country): CountryValidation&#xa;+validateEnrichmentData(enrichmentData, errors): void&#xa;+validateXMLStructure(parsedXml, errors): void&#xa;+publishToKafka(message): Promise&lt;boolean&gt;&#xa;+healthCheck(): Promise&lt;HealthStatus&gt;" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="ddiService">

<mxGeometry y="88" width="320" height="192" as="geometry" />

</mxCell>

<!-- ValidationHandler -->

<mxCell id="valHandler" value="ValidationHandler" style="swimlane;fontStyle=1;align=center;verticalAlign=top;childLayout=stackLayout;horizontal=1;startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;resizeLast=0;collapsible=1;marginBottom=0;" vertex="1" parent="1">

<mxGeometry x="600" y="100" width="350" height="140" as="geometry" />

</mxCell>

<mxCell id="valHandlerFields" value="-validationService: DDIValidationService" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="valHandler">

<mxGeometry y="26" width="350" height="26" as="geometry" />

</mxCell>

<mxCell id="valHandlerLine" value="" style="line;strokeWidth=1;fillColor=none;align=left;verticalAlign=middle;spacingTop=-1;spacingLeft=3;spacingRight=3;rotatable=0;labelPosition=right;points=[];portConstraint=eastwest;strokeColor=inherit;" vertex="1" parent="valHandler">

<mxGeometry y="52" width="350" height="8" as="geometry" />

</mxCell>

<mxCell id="valHandlerMethods" value="+validateEnrichedMessage(call, callback): void&#xa;+healthCheck(call, callback): void&#xa;+convertEnrichmentDataFromGrpc(data): EnrichmentData&#xa;+convertValidationResultToGrpc(result): ValidationResult" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="valHandler">

<mxGeometry y="60" width="350" height="80" as="geometry" />

</mxCell>

<!-- KafkaClient -->

<mxCell id="kafkaClient" value="KafkaClient" style="swimlane;fontStyle=1;align=center;verticalAlign=top;childLayout=stackLayout;horizontal=1;startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;resizeLast=0;collapsible=1;marginBottom=0;" vertex="1" parent="1">

<mxGeometry x="80" y="450" width="280" height="130" as="geometry" />

</mxCell>

<mxCell id="kafkaFields" value="-producer: Producer" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="kafkaClient">

<mxGeometry y="26" width="280" height="26" as="geometry" />

</mxCell>

<mxCell id="kafkaLine" value="" style="line;strokeWidth=1;fillColor=none;align=left;verticalAlign=middle;spacingTop=-1;spacingLeft=3;spacingRight=3;rotatable=0;labelPosition=right;points=[];portConstraint=eastwest;strokeColor=inherit;" vertex="1" parent="kafkaClient">

<mxGeometry y="52" width="280" height="8" as="geometry" />

</mxCell>

<mxCell id="kafkaMethods" value="+publishValidatedMessage(message): Promise&lt;boolean&gt;&#xa;+healthCheck(): Promise&lt;HealthStatus&gt;&#xa;+connect(): Promise&lt;void&gt;&#xa;+disconnect(): Promise&lt;void&gt;" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="kafkaClient">

<mxGeometry y="60" width="280" height="70" as="geometry" />

</mxCell>

<!-- XMLParser -->

<mxCell id="xmlParser" value="XMLParser" style="swimlane;fontStyle=1;align=center;verticalAlign=top;childLayout=stackLayout;horizontal=1;startSize=26;horizontalStack=0;resizeParent=1;resizeParentMax=0;resizeLast=0;collapsible=1;marginBottom=0;" vertex="1" parent="1">

<mxGeometry x="420" y="450" width="380" height="110" as="geometry" />

</mxCell>

<mxCell id="xmlLine" value="" style="line;strokeWidth=1;fillColor=none;align=left;verticalAlign=middle;spacingTop=-1;spacingLeft=3;spacingRight=3;rotatable=0;labelPosition=right;points=[];portConstraint=eastwest;strokeColor=inherit;" vertex="1" parent="xmlParser">

<mxGeometry y="26" width="380" height="8" as="geometry" />

</mxCell>

<mxCell id="xmlMethods" value="+parseXML(xmlPayload): any&#xa;+convertToJSON(xmlPayload, enrichmentData, messageId, puid, messageType): any&#xa;+extractFields(parsedXml): MessageFields" style="text;strokeColor=none;fillColor=none;align=left;verticalAlign=top;spacingLeft=4;spacingRight=4;overflow=hidden;rotatable=0;points=[[0,0.5],[1,0.5]];portConstraint=eastwest;" vertex="1" parent="xmlParser">

<mxGeometry y="34" width="380" height="76" as="geometry" />

</mxCell>

<!-- Relationships -->

<mxCell id="rel1" value="" style="endArrow=open;endFill=1;endSize=12;html=1;rounded=0;" edge="1" parent="1" source="ddiService" target="kafkaClient">

<mxGeometry width="160" relative="1" as="geometry">

<mxPoint x="280" y="400" as="sourcePoint" />

<mxPoint x="220" y="440" as="targetPoint" />

</mxGeometry>

</mxCell>

<mxCell id="rel2" value="" style="endArrow=open;endFill=1;endSize=12;html=1;rounded=0;" edge="1" parent="1" source="ddiService" target="xmlParser">

<mxGeometry width="160" relative="1" as="geometry">

<mxPoint x="480" y="400" as="sourcePoint" />

<mxPoint x="580" y="440" as="targetPoint" />

</mxGeometry>

</mxCell>

<mxCell id="rel3" value="" style="endArrow=open;endFill=1;endSize=12;html=1;rounded=0;" edge="1" parent="1" source="valHandler" target="ddiService">

<mxGeometry width="160" relative="1" as="geometry">

<mxPoint x="600" y="200" as="sourcePoint" />

<mxPoint x="520" y="200" as="targetPoint" />

</mxGeometry>

</mxCell>

</root>

</mxGraphModel>

</diagram>

</mxfile>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Request and Response Formats

gRPC Service Definition

syntax = "proto3";

package gpp.g3.ddivalidation;

service DDIValidationService {

rpc ValidateEnrichedMessage(ValidateEnrichedMessageRequest) returns (ValidateEnrichedMessageResponse);

rpc HealthCheck(HealthCheckRequest) returns (HealthCheckResponse);

}

ValidateEnrichedMessageRequest

message ValidateEnrichedMessageRequest {

string message\_id = 1; // UUID for tracking

string puid = 2; // G3I identifier

string message\_type = 3; // PACS008, PACS007, PACS003

string enriched\_xml\_payload = 4; // Enriched XML from processor

EnrichmentData enrichment\_data = 5; // Account enrichment data

int64 timestamp = 6; // Processing timestamp

map<string, string> metadata = 7; // Additional context

}

ValidateEnrichedMessageResponse

message ValidateEnrichedMessageResponse {

string message\_id = 1; // Echo back UUID

string puid = 2; // Echo back G3I identifier

bool success = 3; // Validation success flag

string error\_message = 4; // Error details if failed

ValidationResult validation\_result = 5; // Detailed validation results

string json\_payload = 6; // Converted JSON payload

bool kafka\_published = 7; // Kafka publishing status

int64 processed\_at = 8; // Processing completion time

string next\_service = 9; // Next service identifier

}

ValidationResult

message ValidationResult {

bool is\_valid = 1; // Overall validation status

repeated ValidationError errors = 2; // List of validation errors

CurrencyValidation currency\_validation = 3; // Currency validation details

CountryValidation country\_validation = 4; // Country validation details

map<string, string> validation\_metadata = 5; // Additional validation data

}

EnrichmentData

message EnrichmentData {

string received\_acct\_id = 1; // Original account ID

int32 lookup\_status\_code = 2; // Lookup status (200=success)

string lookup\_status\_desc = 3; // Status description

string normalized\_acct\_id = 4; // Normalized account ID

string matched\_acct\_id = 5; // Matched account ID

string partial\_match = 6; // Partial match flag (Y/N)

string is\_physical = 7; // Physical account flag (Y/N)

PhysicalAcctInfo physical\_acct\_info = 8; // Account details

string auth\_method = 9; // Authentication method

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Rules and Validation Logic

Singapore Market Validation Rules

| Rule Type | Validation Logic | Expected Value |

|-----------|------------------|----------------|

**| Currency | Must be SGD for Singapore market | SGD |**

**| Country | Must be SG for local processing | SG |**

**| Account Format | Validates account ID formats | Various patterns |**

**| Amount | Validates transaction amounts and limits | Positive numbers |**

**| Enrichment Data | Validates completeness of enrichment | Required fields present |**

Validation Error Codes

| Error Code | Description | Severity |

|------------|-------------|----------|

| INVALID\_CURRENCY | Non-SGD currency detected | ERROR |

| INVALID\_COUNTRY | Non-SG country detected | ERROR |

| MISSING\_ENRICHMENT | Required enrichment data missing | ERROR |

| INVALID\_XML\_STRUCTURE | Malformed XML structure | ERROR |

| VALIDATION\_ERROR | General validation failure | ERROR |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Configuration

Environment Variables

# gRPC Configuration

GRPC\_PORT=50053

SERVICE\_NAME=fast-ddi-validation-service

# Market Configuration

EXPECTED\_CURRENCY=SGD

EXPECTED\_COUNTRY=SG

TIMEZONE=Asia/Singapore

# Kafka Configuration

KAFKA\_BROKERS=localhost:9092

KAFKA\_TOPIC=validated-messages

KAFKA\_CLIENT\_ID=fast-ddi-validation-service

# Processing Configuration

VALIDATION\_TIMEOUT\_MS=5000

MAX\_RETRY\_ATTEMPTS=3

RETRY\_BACKOFF\_MS=1000

# Test Configuration

USE\_TEST\_MODE=false

ENVIRONMENT=development

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Database Schema

**Note: This service does not maintain persistent data storage. It operates as a stateless validation and transformation service.**

Kafka Topic Schema

#### Topic: `validated-messages`

{

"messageId": "string",

"puid": "string",

"messageType": "string",

"jsonPayload": {

"messageId": "string",

"puid": "string",

"messageType": "string",

"enrichedXmlPayload": "string",

"enrichmentData": "object",

"extractedFields": {

"cdtrAcct": "string",

"amount": "string",

"currency": "string",

"country": "string"

},

"processedAt": "string",

"sourceService": "fast-ddi-validation-service"

},

"enrichmentData": "object",

"validationResult": "object",

"timestamp": "number"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Service Integration

Upstream Services

* \*\*fast-inwd-processor-service\*\* (Port 50052): Provides enriched messages for validation

Downstream Services

* \*\*fast-orchestrator-service\*\* (Port 3004): Consumes validated messages via Kafka

Message Flow

fast-inwd-processor-service (PACS.003)

↓ (gRPC ValidateEnrichedMessage)

fast-ddi-validation-service

↓ (Kafka: validated-messages)

fast-orchestrator-service

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error Handling

Error Response Structure

{

"messageId": "uuid",

"puid": "G3I123456789",

"success": false,

"errorMessage": "Singapore market validation failed",

"validationResult": {

"isValid": false,

"errors": [

{

"field": "currency",

"errorCode": "INVALID\_CURRENCY",

"errorMessage": "Invalid currency: USD, expected SGD",

"severity": "ERROR"

}

],

"currencyValidation": {

"isValid": false,

"expectedCurrency": "SGD",

"validationMessage": "Currency validation failed"

},

"countryValidation": {

"isValid": false,

"expectedCountry": "SG",

"validationMessage": "Country validation failed"

}

},

"kafkaPublished": false,

"processedAt": 1640995200000

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Characteristics

Service Performance Metrics

* \*\*Average Response Time\*\*: < 300ms for validation processing
* \*\*Throughput\*\*: 1000+ messages per second
* \*\*Success Rate\*\*: 99.9% for valid Singapore messages
* \*\*Kafka Publishing\*\*: < 100ms additional latency

Resource Requirements

* \*\*CPU\*\*: Low to moderate (validation logic)
* \*\*Memory\*\*: 512MB - 1GB (XML parsing and transformation)
* \*\*Network\*\*: High (Kafka publishing)
* \*\*Storage\*\*: Minimal (stateless service)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Monitoring and Health Checks

Health Check Endpoint

rpc HealthCheck(HealthCheckRequest) returns (HealthCheckResponse);

Health Check Response

{

"status": "SERVING", // SERVING, NOT\_SERVING, UNKNOWN

"message": "Validation service is healthy",

"timestamp": 1640995200000

}

Monitoring Metrics

* Validation success/failure rates
* Processing latency per message type
* Kafka publishing success rates
* Error distribution by validation rule
* Service uptime and availability

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Security Considerations

Data Protection

* No sensitive data persistence
* Secure gRPC communication
* Kafka message encryption in transit
* Input validation and sanitization

Access Control

* Service-to-service authentication
* Network-level security (VPC/firewall rules)
* Role-based access for monitoring

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Deployment Notes

Dependencies

* Kafka cluster availability
* Network connectivity to orchestrator service
* gRPC port 50053 accessibility

Scaling Considerations

* Horizontal scaling supported (stateless)
* Kafka partitioning for load distribution
* Load balancing across service instances

Configuration Management

* Environment-specific configuration
* Feature flags for test mode
* Market-specific validation rules

Fast Inward Processor Service - Design Document

Overview

**The Fast Inward Processor Service is the central orchestration hub of the Singapore G3 Payment Platform. It coordinates account lookup, reference data retrieval, and message enrichment for inward payment processing. This gRPC service serves as the primary integration point between multiple services and implements intelligent routing based on message types.**

Key Responsibilities

* Orchestrate account lookup and reference data retrieval
* Enrich PACS messages with account information and authentication methods
* Implement intelligent routing (PACS.003 → Validation, PACS.008/007 → Direct Kafka)
* Coordinate with multiple downstream services
* Manage message transformation and enrichment workflows

Service Details

* \*\*Service Type\*\*: gRPC Service (Central Hub)
* \*\*Port\*\*: 50052
* \*\*Package\*\*: `gpp.g3.inwdprocessor`
* \*\*Technology Stack\*\*: TypeScript, gRPC, Kafka
* \*\*Role\*\*: Central orchestration hub for message enrichment

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sequence Diagram

sequenceDiagram

participant RH as fast-requesthandler-service

participant IWP as fast-inwd-processor-service

participant AL as fast-accountlookup-service

participant RD as fast-referencedata-service

participant VAL as fast-ddi-validation-service

participant KAFKA as Kafka Broker

RH->>IWP: ProcessMessage(PACS.008)

Note over IWP: Extract CdtrAcct from XML

IWP->>AL: LookupAccount(cdtrAcctId)

AL->>IWP: AccountLookupResponse(acctSys=VAM)

IWP->>RD: LookupAuthMethod(acctSys, acctId)

RD->>IWP: AuthMethodResponse(authMethod=GROUPLIMIT)

Note over IWP: Create enriched XML payload

Note over IWP: Determine routing (DIRECT\_KAFKA)

IWP->>KAFKA: Publish to enriched-messages topic

IWP->>RH: ProcessorResponse(success=true)

Note over IWP: Alternative flow for PACS.003

RH->>IWP: ProcessMessage(PACS.003)

Note over IWP: Determine routing (VALIDATION\_SERVICE)

IWP->>VAL: ValidateEnrichedMessage(enrichedPayload)

VAL->>IWP: ValidationResponse(success=true)

IWP->>RH: ProcessorResponse(success=true)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class Diagram

classDiagram

class InwdProcessorService {

-accountLookupClient: AccountLookupClient

-referenceDataClient: ReferenceDataClient

-validationClient: ValidationClient

-kafkaClient: KafkaClient

-useMockMode: boolean

+enrichMessage(request): Promise<EnrichmentResponse>

+performEnrichment(request, cdtrAcct): Promise<EnrichmentResult>

+determineRouting(messageType): string

+routeToValidationService(request, payload, data): Promise<Response>

+routeToKafkaDirectly(request, payload, data): Promise<Response>

+createJSONPayload(request, payload, data): any

+createEnrichedXML(xmlPayload, enrichmentData): string

+getMarketConfig(metadata): MarketConfig

}

class InwdProcessorHandler {

-enrichmentService: InwdProcessorService

+enrichMessage(call, callback): void

+healthCheck(call, callback): void

+convertEnrichmentDataToGrpc(data): ProcessorData

}

class AccountLookupClient {

-client: grpc.Client

+lookupAccount(request): Promise<AccountLookupResponse>

+healthCheck(): Promise<HealthStatus>

}

class ReferenceDataClient {

+lookupAuthMethod(request): Promise<AuthMethodResponse>

+healthCheck(): Promise<HealthStatus>

}

class ValidationClient {

-client: grpc.Client

+validateEnrichedMessage(request): Promise<ValidationResponse>

}

class KafkaClient {

-producer: Producer

+publishEnrichedMessage(message): Promise<boolean>

+connect(): Promise<void>

+disconnect(): Promise<void>

}

InwdProcessorService --> AccountLookupClient

InwdProcessorService --> ReferenceDataClient

InwdProcessorService --> ValidationClient

InwdProcessorService --> KafkaClient

InwdProcessorHandler --> InwdProcessorService

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Request and Response Formats

gRPC Service Definition

syntax = "proto3";

package gpp.g3.inwdprocessor;

service InwdProcessorService {

rpc ProcessMessage(ProcessorRequest) returns (ProcessorResponse);

rpc HealthCheck(HealthCheckRequest) returns (HealthCheckResponse);

}

ProcessorRequest

message ProcessorRequest {

string message\_id = 1; // UUID from request handler

string puid = 2; // G3I identifier

string message\_type = 3; // PACS008, PACS007, PACS003, CAMT053, etc.

string xml\_payload = 4; // The validated XML payload

map<string, string> metadata = 5; // Additional context data (market info)

int64 timestamp = 6; // Original processing timestamp

}

ProcessorResponse

message ProcessorResponse {

string message\_id = 1; // Echo back the UUID

string puid = 2; // Echo back the G3I identifier

bool success = 3; // Whether processing was successful

string enriched\_payload = 4; // The enriched XML payload

string error\_message = 5; // Error details if success = false

ProcessorData enrichment\_data = 6; // Enrichment data from account lookup

int64 processed\_at = 7; // When processing completed

string next\_service = 8; // Next service in pipeline

}

ProcessorData (Enrichment Data)

message ProcessorData {

string received\_acct\_id = 1; // Original CdtrAcct ID

int32 lookup\_status\_code = 2; // 200 for success

string lookup\_status\_desc = 3; // Success description

string normalized\_acct\_id = 4; // Normalized account ID

string matched\_acct\_id = 5; // Matched account ID

string partial\_match = 6; // Y or N

string is\_physical = 7; // Y or N

PhysicalAccountInfo physical\_acct\_info = 8; // Complex account information

string auth\_method = 9; // Authentication method

}

PhysicalAccountInfo

message PhysicalAccountInfo {

string acct\_id = 1; // Account ID

string acct\_sys = 2; // Account system (MDZ, VAM, MEPS, etc.)

string acct\_group = 3; // Account group (varies by market)

string country = 4; // Country code (SG, MY, TH, etc.)

string branch\_id = 5; // Branch ID (nullable)

AccountAttributes acct\_attributes = 6;

AccountOpsAttributes acct\_ops\_attributes = 7;

string bicfi = 8; // Bank identifier (market-specific)

string currency\_code = 9; // Currency code (SGD, MYR, THB, USD, EUR)

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Logic and Routing Rules

Message Type Routing

| Message Type | Routing Decision | Destination |

|--------------|------------------|-------------|

**| PACS.003 | VALIDATION\_SERVICE | fast-ddi-validation-service → Kafka |**

**| PACS.008 | DIRECT\_KAFKA | Direct to enriched-messages topic |**

**| PACS.007 | DIRECT\_KAFKA | Direct to enriched-messages topic |**

**| Default | VALIDATION\_SERVICE | fast-ddi-validation-service → Kafka |**

Account System Detection

// Account System Logic (via Account Lookup Service)

determineAccountSystem(accountId: string): string {

// VAM accounts: accounts starting with 999 or containing VAM

if (accountId.startsWith('999') || accountId.includes('VAM')) {

return 'VAM';

}

// All other accounts use MDZ

return 'MDZ';

}

Authentication Method Rules

// Authentication Method Logic (via Reference Data Service)

determineAuthMethod(accountId: string): string {

if (accountId.startsWith('999') || accountId.startsWith('VAM')) {

return 'GROUPLIMIT'; // VAM accounts require group limits

}

if (accountId.startsWith('888') || accountId.includes('CORP')) {

return 'AFPTHENLIMIT'; // Corporate accounts

}

if (accountId.startsWith('777') || accountId.includes('PRIV')) {

return 'AFPONLY'; // Private accounts

}

return 'AFPONLY'; // Default

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Enrichment Data Structure

Complete Enrichment Response

{

"receivedAcctId": "999888777666",

"lookupStatusCode": 200,

"lookupStatusDesc": "Account lookup successful",

"normalizedAcctId": "999888777666",

"matchedAcctId": "999888777666",

"partialMatch": "N",

"isPhysical": "Y",

"authMethod": "GROUPLIMIT",

"physicalAcctInfo": {

"acctId": "999888777666",

"acctSys": "VAM",

"acctGroup": "RETAIL",

"country": "SG",

"branchId": "001",

"acctAttributes": {

"acctType": "Physical",

"acctCategory": "SAVINGS",

"acctPurpose": "PERSONAL\_BANKING"

},

"acctOpsAttributes": {

"isActive": "Yes",

"acctStatus": "Active",

"openDate": "15/03/2020",

"expiryDate": "31/12/2025",

"restraints": {

"stopAll": "N",

"stopDebits": "N",

"stopCredits": "N",

"stopAtm": "N",

"stopEftPos": "N",

"stopUnknown": "N",

"warnings": []

}

},

"bicfi": "ANZBSG3MXXX",

"currencyCode": "SGD"

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Integration Patterns

Service Integration Flow

1. Request Handler → Inward Processor (ProcessMessage)

2. Inward Processor → Account Lookup (LookupAccount)

3. Account Lookup → Inward Processor (EnrichmentData with acctSys)

4. Inward Processor → Reference Data (LookupAuthMethod)

5. Reference Data → Inward Processor (AuthMethod)

6. Inward Processor → Create Enriched XML

7. Inward Processor → Route based on message type:

- PACS.003 → Validation Service → Kafka

- PACS.008/007 → Direct Kafka

Kafka Topic Strategy

#### Topic: `enriched-messages` (PACS.008, PACS.007)

{

"messageId": "uuid",

"puid": "G3I123456789",

"messageType": "PACS.008",

"jsonPayload": {

"messageId": "uuid",

"puid": "G3I123456789",

"messageType": "PACS.008",

"enrichedXmlPayload": "enriched-xml-content",

"enrichmentData": "enrichment-object",

"extractedFields": {

"cdtrAcct": "999888777666",

"amount": "1000.00",

"currency": "SGD",

"country": "SG"

},

"processedAt": "2024-01-01T10:00:00Z",

"sourceService": "fast-enrichment-service"

},

"enrichmentData": "enrichment-object",

"timestamp": 1640995200000,

"sourceService": "fast-enrichment-service"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Configuration

Environment Variables

# gRPC Configuration

GRPC\_PORT=50052

SERVICE\_NAME=fast-inwd-processor-service

# Market Configuration

COUNTRY=SG

DEFAULT\_CURRENCY=SGD

TIMEZONE=Asia/Singapore

# Downstream Service URLs

ACCOUNT\_LOOKUP\_SERVICE\_URL=localhost:50059

REFERENCE\_DATA\_SERVICE\_URL=localhost:50060

VALIDATION\_SERVICE\_URL=localhost:50053

# Kafka Configuration

KAFKA\_BROKERS=localhost:9092

KAFKA\_CLIENT\_ID=fast-inwd-processor-service

ENRICHED\_MESSAGES\_TOPIC=enriched-messages

# Processing Configuration

LOOKUP\_TIMEOUT\_MS=3000

MAX\_RETRY\_ATTEMPTS=3

RETRY\_BACKOFF\_MS=1000

USE\_MOCK\_MODE=false

# Test Configuration

NODE\_ENV=development

ENVIRONMENT=development

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Database Schema

**Note: This service does not maintain persistent data storage. It operates as a stateless orchestration service that coordinates between multiple data sources.**

Mock Data Storage (Test Mode Only)

interface MockAccountData {

accountId: string;

accountSystem: 'VAM' | 'MDZ' | 'MEPS';

authMethod: 'GROUPLIMIT' | 'AFPTHENLIMIT' | 'AFPONLY';

country: string;

currency: string;

isActive: boolean;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Service Dependencies

Required Services

| Service | Port | Purpose | Status |

|---------|------|---------|--------|

**| fast-accountlookup-service | 50059 | Account information retrieval | Required |**

**| fast-referencedata-service | 50060 | Authentication method lookup | Required |**

**| fast-ddi-validation-service | 50053 | Message validation (PACS.003) | Required |**

**| Kafka Broker | 9092 | Message publishing (PACS.008/007) | Required |**

Optional Integrations

| Service | Port | Purpose | Fallback |

|---------|------|---------|----------|

**| Monitoring Service | Various | Health and metrics | Local logging |**

**| Configuration Service | Various | Dynamic configuration | Environment variables |**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error Handling

Error Response Structure

{

"messageId": "uuid",

"puid": "G3I123456789",

"success": false,

"errorMessage": "Account lookup failed: Service unavailable",

"enrichmentData": null,

"processedAt": 1640995200000,

"nextService": "",

"errorCode": "ACCOUNT\_LOOKUP\_ERROR",

"errorDetails": {

"service": "fast-accountlookup-service",

"operation": "LookupAccount",

"cdtrAcctId": "123456789",

"retryAttempts": 3,

"lastError": "Connection timeout"

}

}

Error Categories

| Error Type | Code | Description | Recovery Action |

|------------|------|-------------|-----------------|

**| Account Lookup Errors | LOOKUP\_ERROR | Account service unavailable | Retry with exponential backoff |**

**| Reference Data Errors | REFDATA\_ERROR | Auth method lookup failed | Use default auth method |**

**| Validation Errors | VALIDATION\_ERROR | Validation service failed | Return error to client |**

**| Kafka Errors | KAFKA\_ERROR | Publishing failed | Retry publishing |**

**| XML Processing Errors | XML\_ERROR | Invalid XML structure | Return validation error |**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Characteristics

Service Performance Metrics

* \*\*Average Response Time\*\*: 150-300ms (including downstream calls)
* \*\*Account Lookup Time\*\*: ~110ms average
* \*\*Reference Data Time\*\*: ~50ms average
* \*\*XML Processing Time\*\*: ~20ms average
* \*\*Kafka Publishing Time\*\*: ~30ms average

Throughput Specifications

* \*\*Target Throughput\*\*: 1000+ messages per second
* \*\*Concurrent Requests\*\*: 100+ simultaneous
* \*\*Resource Usage\*\*: 512MB-1GB memory, moderate CPU

SLA Requirements

* \*\*Availability\*\*: 99.9% uptime
* \*\*Response Time\*\*: 95th percentile < 500ms
* \*\*Error Rate\*\*: < 0.1% for valid requests

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Monitoring and Health Checks

Health Check Implementation

async healthCheck(): Promise<HealthCheckResponse> {

const dependencies = await Promise.allSettled([

this.accountLookupClient.healthCheck(),

this.referenceDataClient.healthCheck(),

this.kafkaClient.healthCheck()

]);

const overallHealth = dependencies.every(d => d.status === 'fulfilled');

return {

status: overallHealth ? 'SERVING' : 'NOT\_SERVING',

message: overallHealth ? 'All dependencies healthy' : 'Some dependencies unhealthy',

timestamp: Date.now(),

dependencies: {

accountLookup: dependencies[0].status,

referenceData: dependencies[1].status,

kafka: dependencies[2].status

}

};

}

Monitoring Metrics

* \*\*Message Processing Rates\*\*: By message type and routing decision
* \*\*Downstream Service Latency\*\*: Response times for each dependency
* \*\*Error Rates\*\*: By error type and downstream service
* \*\*Enrichment Success Rates\*\*: Successful vs failed enrichments
* \*\*Routing Distribution\*\*: VALIDATION\_SERVICE vs DIRECT\_KAFKA percentages

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Security Considerations

Data Security

* No persistent sensitive data storage
* Secure gRPC communication with downstream services
* Input validation and sanitization
* XML parsing security (XXE protection)

Access Control

* Service-to-service authentication via gRPC metadata
* Network segmentation for internal services
* Role-based access for monitoring and management

Data Privacy

* Message data encrypted in transit
* No logging of sensitive payment information
* Compliance with PCI DSS requirements

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Deployment and Scaling

Deployment Requirements

* \*\*Platform\*\*: Kubernetes or container orchestration
* \*\*Resources\*\*: 1-2 CPU cores, 1-2GB RAM per instance
* \*\*Network\*\*: Access to all downstream services and Kafka
* \*\*Storage\*\*: Minimal (stateless service)

Scaling Strategy

* \*\*Horizontal Scaling\*\*: Multiple instances behind load balancer
* \*\*Auto-scaling\*\*: Based on CPU usage and request latency
* \*\*Circuit Breakers\*\*: Prevent cascade failures
* \*\*Bulkhead Pattern\*\*: Isolate different message types

High Availability

* \*\*Multi-zone Deployment\*\*: Distribute across availability zones
* \*\*Health Checks\*\*: Kubernetes liveness and readiness probes
* \*\*Graceful Degradation\*\*: Continue with mock data if services unavailable
* \*\*Disaster Recovery\*\*: Rapid service restart and dependency reconnection

Fast Account Lookup Service - Design Document

Overview

**The Fast Account Lookup Service is a gRPC-based microservice that provides account lookup and enrichment capabilities for PACS message processing in the Singapore G3 Payment Platform. It serves as the primary source for account system detection (VAM/MDZ) and comprehensive account enrichment data.**

Key Responsibilities

* Lookup account information based on creditor account IDs
* Determine account system (VAM, MDZ, MEPS, FAST) based on account patterns
* Provide comprehensive account enrichment data
* Support Singapore banking standards and compliance
* Generate realistic mock data for development and testing

Service Details

* \*\*Service Type\*\*: gRPC Service
* \*\*Port\*\*: 50059
* \*\*Package\*\*: `gpp.g3.accountlookup`
* \*\*Technology Stack\*\*: TypeScript, gRPC
* \*\*Implementation\*\*: Currently stubbed with intelligent mock data

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sequence Diagram

sequenceDiagram

participant Client as fast-inwd-processor-service

participant AL as fast-accountlookup-service

participant DB as Account Database

participant Mock as Mock Data Generator

Client->>AL: LookupAccount(cdtrAcctId)

Note over AL: Validate request parameters

alt Production Mode

AL->>DB: Query account database

DB->>AL: Return account details

else Stub Mode (Current)

AL->>Mock: Generate mock account data

Mock->>AL: Return enrichment data

end

Note over AL: Determine account system (VAM/MDZ)

Note over AL: Build comprehensive enrichment data

Note over AL: Apply Singapore banking standards

AL->>Client: AccountLookupResponse(enrichmentData)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class Diagram

classDiagram

class AccountLookupService {

+lookupAccount(request): Promise<AccountLookupResponse>

+healthCheck(): Promise<HealthStatus>

+validateRequest(request): string|null

+generateEnrichmentData(request): EnrichmentData

+handleSimulatedError(request): AccountLookupResponse

+createErrorResponse(request, code, message): AccountLookupResponse

}

class AccountLookupHandler {

-accountLookupService: AccountLookupService

+lookupAccount(call, callback): void

+healthCheck(call, callback): void

+getServiceInfo(call, callback): void

+validateLookupAccountRequest(request): string|null

+convertEnrichmentDataToGrpc(data): EnrichmentData

}

class MockDataGenerator {

+generateSingaporeAccountData(accountId): EnrichmentData

+simulateDelay(delayMs): Promise<void>

+shouldSimulateError(accountId): boolean

+createPhysicalAccountInfo(accountId, accountType): PhysicalAccountInfo

+generateAccountAttributes(accountType): AccountAttributes

+generateOperationalAttributes(): AccountOpsAttributes

}

class AccountUtils {

+getAccountSystem(accountId): string

+getAccountType(accountId): string

+normalizeAccountId(accountId): string

+shouldSimulateError(accountId): boolean

+generateBranchId(): string

+formatDate(date): string

}

AccountLookupService --> MockDataGenerator

AccountLookupService --> AccountUtils

AccountLookupHandler --> AccountLookupService

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Request and Response Formats

gRPC Service Definition

syntax = "proto3";

package gpp.g3.accountlookup;

service AccountLookupService {

rpc LookupAccount(AccountLookupRequest) returns (AccountLookupResponse);

rpc HealthCheck(HealthCheckRequest) returns (HealthCheckResponse);

rpc GetServiceInfo(ServiceInfoRequest) returns (ServiceInfoResponse);

}

AccountLookupRequest

message AccountLookupRequest {

string message\_id = 1; // UUID for tracking

string puid = 2; // G3I identifier

string cdtr\_acct\_id = 3; // CdtrAcct ID to lookup

string message\_type = 4; // PACS message type for context

map<string, string> metadata = 5; // Additional lookup context

int64 timestamp = 6; // Request timestamp

}

AccountLookupResponse

message AccountLookupResponse {

string message\_id = 1; // Echo back UUID

string puid = 2; // Echo back G3I identifier

bool success = 3; // Whether lookup was successful

string error\_message = 4; // Error details if success = false

string error\_code = 5; // Categorized error code

EnrichmentData enrichment\_data = 6; // Full enrichment data

int64 processed\_at = 7; // When lookup completed

string lookup\_source = 8; // Source of lookup (STUB, CACHE, DATABASE)

}

EnrichmentData

message EnrichmentData {

string received\_acct\_id = 1; // Original CdtrAcct ID

int32 lookup\_status\_code = 2; // 200 for success, 404 for not found

string lookup\_status\_desc = 3; // Human-readable status description

string normalized\_acct\_id = 4; // Normalized account ID

string matched\_acct\_id = 5; // Matched account ID

string partial\_match = 6; // Y or N

string is\_physical = 7; // Y or N

PhysicalAccountInfo physical\_acct\_info = 8; // Complex account information

}

PhysicalAccountInfo

message PhysicalAccountInfo {

string acct\_id = 1; // Account ID

string acct\_sys = 2; // Account system (VAM, MDZ, FAST, MEPS)

string acct\_group = 3; // Account group (SGB, etc.)

string country = 4; // Country (SG)

string branch\_id = 5; // Branch ID (nullable)

AccountAttributes acct\_attributes = 6;

AccountOpsAttributes acct\_ops\_attributes = 7;

string bicfi = 8; // Bank identifier (ANZBSG3MXXX)

string currency\_code = 9; // Currency (SGD)

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Logic Implementation

Account System Detection Logic

static getAccountSystem(accountId: string): string {

const normalized = this.normalizeAccountId(accountId);

// VAM accounts: accounts starting with 999 or containing VAM

if (normalized.startsWith('999') || normalized.includes('VAM')) {

return 'VAM';

}

// FAST system for utility accounts

if (normalized.startsWith('SP') || normalized.includes('UTIL')) {

return 'FAST';

}

// MEPS for government/corporate accounts

if (normalized.startsWith('GOVT') || normalized.startsWith('CORP')) {

return 'MEPS';

}

// Default to MDZ for all other accounts

return 'MDZ';

}

Account Type Detection

static getAccountType(accountId: string): string {

const normalized = accountId.trim().toUpperCase();

if (normalized.startsWith('CORP')) {

return 'Corporate';

}

if (normalized.startsWith('GOVT')) {

return 'Government';

}

if (normalized.startsWith('UTIL') || normalized.startsWith('SP')) {

return 'Utility';

}

return 'Physical';

}

Mock Data Generation

generateEnrichmentData(request: AccountLookupRequest): EnrichmentData {

const accountId = request.cdtrAcctId;

const accountSystem = AccountUtils.getAccountSystem(accountId);

const accountType = AccountUtils.getAccountType(accountId);

return {

receivedAcctId: accountId,

lookupStatusCode: 200,

lookupStatusDesc: 'Success',

normalizedAcctId: AccountUtils.normalizeAccountId(accountId),

matchedAcctId: accountId,

partialMatch: 'N',

isPhysical: 'Y',

physicalAcctInfo: this.createPhysicalAccountInfo(accountId, accountSystem, accountType)

};

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Account Types and Business Rules

Singapore Account System Mapping

| Account Pattern | Account System | Use Case |

|-----------------|----------------|----------|

**| 999 or VAM\* | VAM | High-value accounts, require group limits |**

**| SP or UTIL\* | FAST | Utility payments, instant settlement |**

**| GOVT\* | MEPS | Government accounts, regulatory compliance |**

**| CORP\* | MEPS | Corporate accounts, bulk processing |**

**| Other patterns | MDZ | Standard retail banking |**

Account Type Classifications

| Account Type | Description | System Preference | Special Handling |

|--------------|-------------|-------------------|------------------|

**| Physical | Standard customer accounts | MDZ | Standard validation |**

**| Corporate | Business accounts | MEPS | Enhanced due diligence |**

**| Government | Government entity accounts | MEPS | Regulatory reporting |**

**| Utility | Service provider accounts | FAST | Instant settlement |**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sample Response Data

Successful Lookup Response

{

"messageId": "550e8400-e29b-41d4-a716-446655440000",

"puid": "G3I1234567890123",

"success": true,

"enrichmentData": {

"receivedAcctId": "999888777666",

"lookupStatusCode": 200,

"lookupStatusDesc": "Success",

"normalizedAcctId": "999888777666",

"matchedAcctId": "999888777666",

"partialMatch": "N",

"isPhysical": "Y",

"physicalAcctInfo": {

"acctId": "999888777666",

"acctSys": "VAM",

"acctGroup": "SGB",

"country": "SG",

"branchId": "001",

"acctAttributes": {

"acctType": "Physical",

"acctCategory": "SAVINGS",

"acctPurpose": "PERSONAL\_BANKING"

},

"acctOpsAttributes": {

"isActive": "Yes",

"acctStatus": "Active",

"openDate": "15/03/2020",

"expiryDate": "31/12/2025",

"restraints": {

"stopAll": "N",

"stopDebits": "N",

"stopCredits": "N",

"stopAtm": "N",

"stopEftPos": "N",

"stopUnknown": "N",

"warnings": []

}

},

"bicfi": "ANZBSG3MXXX",

"currencyCode": "SGD"

}

},

"processedAt": 1640995200000,

"lookupSource": "STUB"

}

Error Response Example

{

"messageId": "550e8400-e29b-41d4-a716-446655440000",

"puid": "G3I1234567890123",

"success": false,

"errorMessage": "Account not found in system",

"errorCode": "LOOKUP\_ACCOUNT\_NOT\_FOUND\_002",

"enrichmentData": null,

"processedAt": 1640995200000,

"lookupSource": "STUB"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error Handling

Error Scenarios and Codes

| Scenario | Error Code | HTTP Status | Description |

|----------|------------|-------------|-------------|

**| Account Not Found | LOOKUP\_ACCOUNT\_NOT\_FOUND\_002 | 404 | Account ID not in system |**

**| Invalid Request | LOOKUP\_INVALID\_REQUEST\_001 | 400 | Missing required parameters |**

**| Service Unavailable | LOOKUP\_SERVICE\_ERROR\_003 | 500 | Internal service error |**

**| Timeout | LOOKUP\_TIMEOUT\_004 | 504 | Request processing timeout |**

**| Rate Limited | LOOKUP\_RATE\_LIMITED\_005 | 429 | Too many requests |**

Error Simulation (Test Mode)

// Test accounts for error simulation

const ERROR\_SIMULATION\_PATTERNS = {

'NOTFOUND': { code: 'LOOKUP\_ACCOUNT\_NOT\_FOUND\_002', status: 404 },

'ERROR': { code: 'LOOKUP\_SERVICE\_ERROR\_003', status: 500 },

'TIMEOUT': { code: 'LOOKUP\_TIMEOUT\_004', status: 504 },

'INACTIVE': { code: 'LOOKUP\_ACCOUNT\_INACTIVE\_006', status: 200 }

};

shouldSimulateError(accountId: string): boolean {

return Object.keys(ERROR\_SIMULATION\_PATTERNS)

.some(pattern => accountId.toUpperCase().includes(pattern));

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Configuration

Environment Variables

# gRPC Configuration

GRPC\_PORT=50059

SERVICE\_NAME=fast-accountlookup-service

# Service Configuration

LOG\_LEVEL=info

ENVIRONMENT=development

IS\_STUBBED=true

COUNTRY=SG

DEFAULT\_CURRENCY=SGD

TIMEZONE=Asia/Singapore

DEFAULT\_BANK\_CODE=ANZBSG3MXXX

# Mock Data Configuration

MOCK\_SUCCESS\_RATE=0.95

MOCK\_RESPONSE\_DELAY\_MS=100

ENABLE\_ERROR\_SCENARIOS=true

DEFAULT\_ACCOUNT\_TYPE=Physical

# Processing Configuration

LOOKUP\_TIMEOUT\_MS=3000

MAX\_RETRY\_ATTEMPTS=2

RATE\_LIMIT\_REQUESTS\_PER\_MINUTE=1000

# Database Configuration (Future)

DATABASE\_URL=postgresql://localhost:5432/accounts

CACHE\_TTL\_SECONDS=300

ENABLE\_CACHE=true

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Database Schema

**Note: Currently implemented as a stubbed service. Future production implementation will require the following database schema:**

Account Information Table

CREATE TABLE account\_information (

account\_id VARCHAR(50) PRIMARY KEY,

account\_system VARCHAR(20) NOT NULL, -- VAM, MDZ, FAST, MEPS

account\_group VARCHAR(20) NOT NULL, -- SGB, RETAIL, CORPORATE

account\_type VARCHAR(20) NOT NULL, -- Physical, Virtual, Corporate

account\_category VARCHAR(50), -- SAVINGS, CURRENT, CORPORATE

account\_purpose VARCHAR(100), -- PERSONAL\_BANKING, BUSINESS

country VARCHAR(3) NOT NULL DEFAULT 'SG',

currency\_code VARCHAR(3) NOT NULL DEFAULT 'SGD',

bicfi VARCHAR(11) NOT NULL DEFAULT 'ANZBSG3MXXX',

branch\_id VARCHAR(10),

is\_active BOOLEAN NOT NULL DEFAULT true,

account\_status VARCHAR(20) NOT NULL DEFAULT 'Active',

open\_date DATE NOT NULL,

expiry\_date DATE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_account\_system ON account\_information(account\_system);

CREATE INDEX idx\_account\_type ON account\_information(account\_type);

CREATE INDEX idx\_country\_currency ON account\_information(country, currency\_code);

CREATE INDEX idx\_status\_active ON account\_information(account\_status, is\_active);

Account Restraints Table

CREATE TABLE account\_restraints (

account\_id VARCHAR(50) PRIMARY KEY,

stop\_all BOOLEAN DEFAULT false,

stop\_debits BOOLEAN DEFAULT false,

stop\_credits BOOLEAN DEFAULT false,

stop\_atm BOOLEAN DEFAULT false,

stop\_eft\_pos BOOLEAN DEFAULT false,

stop\_unknown BOOLEAN DEFAULT false,

warnings TEXT[],

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (account\_id) REFERENCES account\_information(account\_id)

);

Account Lookup Cache

CREATE TABLE account\_lookup\_cache (

cache\_key VARCHAR(255) PRIMARY KEY,

account\_id VARCHAR(50) NOT NULL,

enrichment\_data JSONB NOT NULL,

cached\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP NOT NULL,

hit\_count INTEGER DEFAULT 0

);

CREATE INDEX idx\_cache\_expiry ON account\_lookup\_cache(expires\_at);

CREATE INDEX idx\_cache\_account ON account\_lookup\_cache(account\_id);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Service Integration

Upstream Services

* \*\*fast-inwd-processor-service\*\* (Port 50052): Primary consumer for account lookups

Integration Pattern

// Called by Inward Processor Service

const lookupRequest: AccountLookupRequest = {

messageId: "uuid",

puid: "G3I123456789",

cdtrAcctId: "999888777666",

messageType: "PACS.008",

metadata: { country: "SG", currency: "SGD" },

timestamp: Date.now()

};

const response = await accountLookupClient.LookupAccount(lookupRequest);

Response Integration

// Response used by Inward Processor for enrichment

if (response.success) {

const enrichmentData = response.enrichmentData;

const accountSystem = enrichmentData.physicalAcctInfo.acctSys; // VAM, MDZ, etc.

// Use account system for downstream routing decisions

if (accountSystem === 'VAM') {

// Route to VAM mediation

} else if (accountSystem === 'MDZ') {

// Route to standard processing

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Characteristics

Current Performance (Stub Mode)

* \*\*Average Response Time\*\*: ~110ms
* \*\*Success Rate\*\*: 100% (configurable via mock settings)
* \*\*Concurrent Requests\*\*: Limited by gRPC server configuration
* \*\*Memory Usage\*\*: Low (stateless mock data generation)

Target Production Performance

* \*\*Response Time\*\*: < 100ms (95th percentile)
* \*\*Throughput\*\*: 5000+ requests per second
* \*\*Cache Hit Rate\*\*: > 80% for frequently accessed accounts
* \*\*Database Query Time\*\*: < 50ms average

Scalability Considerations

* \*\*Horizontal Scaling\*\*: Multiple service instances behind load balancer
* \*\*Caching Strategy\*\*: Redis/Memcached for account data
* \*\*Database Optimization\*\*: Read replicas, connection pooling
* \*\*Circuit Breaker\*\*: Fail fast on database issues

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Monitoring and Health Checks

Health Check Implementation

async healthCheck(): Promise<HealthStatus> {

if (config.isStubbed) {

return {

status: 'SERVING',

message: 'Account lookup service is healthy (stub mode)',

timestamp: Date.now(),

capabilities: [

'account-lookup',

'singapore-banking-data',

'mock-data-generation',

'error-simulation'

]

};

}

// Production health checks

const dbHealth = await this.checkDatabaseHealth();

const cacheHealth = await this.checkCacheHealth();

const isHealthy = dbHealth && cacheHealth;

return {

status: isHealthy ? 'SERVING' : 'NOT\_SERVING',

message: isHealthy ? 'All systems operational' : 'Service degraded',

timestamp: Date.now(),

dependencies: {

database: dbHealth,

cache: cacheHealth

}

};

}

Monitoring Metrics

// Prometheus metrics

const metrics = {

lookupRequests: new Counter({

name: 'accountlookup\_requests\_total',

help: 'Total account lookup requests',

labelNames: ['account\_type', 'account\_system', 'status']

}),

lookupDuration: new Histogram({

name: 'accountlookup\_duration\_seconds',

help: 'Account lookup duration',

buckets: [0.01, 0.05, 0.1, 0.2, 0.5, 1.0]

}),

cacheHitRate: new Gauge({

name: 'accountlookup\_cache\_hit\_rate',

help: 'Cache hit rate percentage'

})

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Security Considerations

Data Protection

* \*\*PII Handling\*\*: Minimal exposure of account holder information
* \*\*Data Encryption\*\*: Encrypt sensitive data in transit and at rest
* \*\*Access Logging\*\*: Log all account access for audit purposes
* \*\*Data Masking\*\*: Mask account details in logs

Input Validation

validateLookupAccountRequest(request: any): string | null {

if (!request.message\_id || request.message\_id.length < 8) {

return 'message\_id must be at least 8 characters long';

}

if (!request.puid || request.puid.length < 8) {

return 'puid must be at least 8 characters long';

}

if (!request.cdtr\_acct\_id || request.cdtr\_acct\_id.length < 3) {

return 'cdtr\_acct\_id must be at least 3 characters long';

}

const validMessageTypes = ['pacs.008.001.10', 'pacs.002.001.15', 'pacs.004.001.12'];

if (!validMessageTypes.includes(request.message\_type)) {

return `message\_type must be one of: ${validMessageTypes.join(', ')}`;

}

return null;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Migration to Production

Phase 1: Database Integration

* Implement PostgreSQL/Cloud Spanner backend
* Create account information tables
* Import Singapore banking account data
* Implement caching layer

Phase 2: Performance Optimization

* Add database connection pooling
* Implement read replicas
* Optimize query performance
* Add comprehensive monitoring

Phase 3: Advanced Features

* Real-time account status updates
* Advanced account matching algorithms
* Multi-market support
* Enhanced error handling

Deployment Strategy

* \*\*Blue-Green Deployment\*\*: Zero-downtime migration
* \*\*Feature Flags\*\*: Gradual rollout of production features
* \*\*Monitoring\*\*: Comprehensive observability during migration
* \*\*Rollback Plan\*\*: Quick revert to stub mode if needed

Fast Reference Data Service - Design Document

Overview

**The Fast Reference Data Service is a gRPC-based microservice that provides authentication method lookup for the Singapore G3 Payment Platform. It determines the appropriate authentication method (GROUPLIMIT, AFPTHENLIMIT, AFPONLY) based on account information and business rules, enabling intelligent routing and processing decisions.**

Key Responsibilities

* Determine authentication methods based on account patterns and business rules
* Provide reference data for risk assessment and limit profiling
* Support multiple account systems (VAM, MDZ, MEPS, FAST)
* Generate comprehensive reference data details for downstream processing
* Implement Singapore-specific business rules and compliance requirements

Service Details

* \*\*Service Type\*\*: gRPC Service
* \*\*Port\*\*: 50060
* \*\*Package\*\*: `gpp.g3.referencedata`
* \*\*Technology Stack\*\*: TypeScript, gRPC
* \*\*Implementation\*\*: Rule-based authentication method determination

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sequence Diagram

sequenceDiagram

participant Client as fast-inwd-processor-service

participant RDS as fast-referencedata-service

participant Rules as Business Rules Engine

participant Cache as Reference Data Cache

Client->>RDS: LookupAuthMethod(acctSys, acctId)

Note over RDS: Validate request parameters

alt Cache Hit

RDS->>Cache: Check cache for auth method

Cache->>RDS: Return cached result

else Cache Miss

RDS->>Rules: Apply business rules

Rules->>Rules: Analyze account ID patterns

Rules->>Rules: Determine risk level

Rules->>Rules: Set limit profile

Rules->>RDS: Return auth method

RDS->>Cache: Store result in cache

end

Note over RDS: Create reference data details

Note over RDS: Apply Singapore compliance rules

RDS->>Client: AuthMethodResponse(authMethod, details)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class Diagram

classDiagram

class ReferenceDataService {

-useMockData: boolean

+lookupAuthMethod(request): Promise<AuthMethodResponse>

+validateRequest(request): string|null

+determineAuthMethod(accountId): string

+createReferenceDataDetails(request, authMethod): ReferenceDataDetails

+determineRiskLevel(authMethod): string

+determineLimitProfile(acctSys, acctGrp, authMethod): string

+createErrorResponse(request, errorMessage): AuthMethodResponse

}

class ReferenceDataHandler {

-referenceDataService: ReferenceDataService

+lookupAuthMethod(call, callback): void

+healthCheck(call, callback): void

+convertRefDataDetailsToGrpc(details): any

}

class BusinessRulesEngine {

+applyAccountPatternRules(accountId): string

+applyLengthBasedRules(accountId): string

+applySystemSpecificRules(acctSys, accountId): string

+getSingaporeComplianceRules(): ComplianceRules

+determineApprovalRequirement(authMethod): boolean

}

class ReferenceDataCache {

-cache: Map<string, CacheEntry>

+get(key): CacheEntry|null

+set(key, value, ttl): void

+invalidate(key): void

+cleanup(): void

}

class Logger {

+logAuthMethodRequest(messageId, puid, acctId, acctSys): void

+logAuthMethodResponse(messageId, authMethod, success, processingTime): void

+logError(messageId, error, context): void

}

ReferenceDataService --> BusinessRulesEngine

ReferenceDataService --> ReferenceDataCache

ReferenceDataService --> Logger

ReferenceDataHandler --> ReferenceDataService

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Request and Response Formats

gRPC Service Definition

syntax = "proto3";

package gpp.g3.referencedata;

service ReferenceDataService {

rpc LookupAuthMethod(AuthMethodRequest) returns (AuthMethodResponse);

rpc HealthCheck(HealthCheckRequest) returns (HealthCheckResponse);

}

AuthMethodRequest

message AuthMethodRequest {

string message\_id = 1; // UUID for tracking

string puid = 2; // G3I identifier

string acct\_sys = 3; // Account system (MDZ, VAM, etc.)

string acct\_grp = 4; // Account group (SGB, etc.)

string acct\_id = 5; // Account ID

string country = 6; // Country code (SG)

string currency\_code = 7; // Currency code (SGD)

map<string, string> metadata = 8; // Additional context

int64 timestamp = 9; // Request timestamp

}

AuthMethodResponse

message AuthMethodResponse {

string message\_id = 1; // Echo back UUID

string puid = 2; // Echo back G3I identifier

bool success = 3; // Whether lookup was successful

string auth\_method = 4; // AFPONLY, AFPTHENLIMIT, GROUPLIMIT

string error\_message = 5; // Error details if success = false

string error\_code = 6; // Categorized error code

ReferenceDataDetails ref\_data\_details = 7; // Additional reference data

int64 processed\_at = 8; // When lookup completed

string lookup\_source = 9; // Source of lookup (STUB, CACHE, DATABASE)

}

ReferenceDataDetails

message ReferenceDataDetails {

string acct\_sys = 1; // Account system

string acct\_grp = 2; // Account group

string acct\_id = 3; // Account ID

string country = 4; // Country

string currency\_code = 5; // Currency code

string auth\_method = 6; // Authentication method

string risk\_level = 7; // Risk level (LOW, MEDIUM, HIGH)

string limit\_profile = 8; // Limit profile identifier

bool requires\_approval = 9; // Whether requires approval

map<string, string> additional\_attributes = 10; // Additional attributes

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Logic and Authentication Rules

Authentication Method Determination Logic

determineAuthMethod(acctId: string): string {

const normalizedAcctId = acctId.trim().toUpperCase();

// Pattern-based authentication method determination

if (normalizedAcctId.startsWith('999') || normalizedAcctId.startsWith('VAM')) {

// VAM accounts typically require group limits

return 'GROUPLIMIT';

}

if (normalizedAcctId.startsWith('888') || normalizedAcctId.includes('CORP')) {

// Corporate accounts use AFP then limits

return 'AFPTHENLIMIT';

}

if (normalizedAcctId.startsWith('777') || normalizedAcctId.includes('PRIV')) {

// Private accounts use AFP only

return 'AFPONLY';

}

// Account ID length based rules

if (normalizedAcctId.length >= 12) {

// Long account IDs typically require group limits

return 'GROUPLIMIT';

}

if (normalizedAcctId.length >= 8) {

// Medium length accounts use AFP then limits

return 'AFPTHENLIMIT';

}

// Default to AFP only for other cases

return 'AFPONLY';

}

Authentication Method Classifications

| Authentication Method | Use Case | Processing Requirements | Limit Checking |

|----------------------|----------|------------------------|-----------------|

**| GROUPLIMIT | High-value VAM accounts, Government accounts | Group-level limit validation required | Mandatory fire & forget |**

**| AFPTHENLIMIT | Corporate accounts, Enhanced validation | AFP validation + individual limits | Individual account limits |**

**| AFPONLY | Standard retail accounts | Anti-Fraud Platform only | No additional limits |**

Risk Level Determination

private determineRiskLevel(authMethod: string): string {

switch (authMethod) {

case 'GROUPLIMIT':

return 'HIGH'; // Requires group-level oversight

case 'AFPTHENLIMIT':

return 'MEDIUM'; // Enhanced validation required

case 'AFPONLY':

return 'LOW'; // Standard processing

default:

return 'MEDIUM'; // Conservative default

}

}

Limit Profile Assignment

private determineLimitProfile(acctSys: string, acctGrp: string, authMethod: string): string {

const base = `${acctSys}\_${acctGrp}`;

switch (authMethod) {

case 'GROUPLIMIT':

return `${base}\_GROUP\_LIMITS`;

case 'AFPTHENLIMIT':

return `${base}\_AFP\_THEN\_LIMITS`;

case 'AFPONLY':

return `${base}\_AFP\_ONLY`;

default:

return `${base}\_STANDARD`;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sample Response Data

Successful Authentication Method Lookup

{

"messageId": "550e8400-e29b-41d4-a716-446655440000",

"puid": "G3I1234567890123",

"success": true,

"authMethod": "GROUPLIMIT",

"refDataDetails": {

"acctSys": "VAM",

"acctGrp": "SGB",

"acctId": "999888777666",

"country": "SG",

"currencyCode": "SGD",

"authMethod": "GROUPLIMIT",

"riskLevel": "HIGH",

"limitProfile": "VAM\_SGB\_GROUP\_LIMITS",

"requiresApproval": true,

"additionalAttributes": {

"evaluatedAt": "2024-01-01T10:00:00Z",

"evaluationRules": "account\_id\_pattern\_based",

"sourceSystem": "fast-referencedata-service",

"complianceLevel": "SINGAPORE\_ENHANCED",

"authenticationTier": "TIER\_3"

}

},

"processedAt": 1640995200000,

"lookupSource": "STUB"

}

Different Authentication Methods by Account Type

#### AFPONLY Example (Standard Retail)

{

"authMethod": "AFPONLY",

"refDataDetails": {

"riskLevel": "LOW",

"limitProfile": "MDZ\_SGB\_AFP\_ONLY",

"requiresApproval": false,

"additionalAttributes": {

"authenticationTier": "TIER\_1",

"processingPriority": "STANDARD"

}

}

}

#### AFPTHENLIMIT Example (Corporate)

{

"authMethod": "AFPTHENLIMIT",

"refDataDetails": {

"riskLevel": "MEDIUM",

"limitProfile": "MDZ\_SGB\_AFP\_THEN\_LIMITS",

"requiresApproval": false,

"additionalAttributes": {

"authenticationTier": "TIER\_2",

"processingPriority": "ENHANCED"

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Rules Engine

Account Pattern Rules

| Pattern | Auth Method | Rationale |

|---------|-------------|-----------|

| 999\* | GROUPLIMIT | VAM high-value accounts |

| VAM\* | GROUPLIMIT | Explicit VAM system accounts |

| 888, CORP\* | AFPTHENLIMIT | Corporate account enhanced validation |

| 777, PRIV\* | AFPONLY | Private account standard processing |

| GOVT\* | GROUPLIMIT | Government accounts require oversight |

| Length ≥ 12 chars | GROUPLIMIT | Complex account structures |

| Length 8-11 chars | AFPTHENLIMIT | Medium complexity accounts |

| Length < 8 chars | AFPONLY | Simple account structures |

Singapore Compliance Rules

const SINGAPORE\_COMPLIANCE\_RULES = {

mandatoryFields: ['acct\_sys', 'acct\_id', 'country', 'currency\_code'],

requiredCurrency: 'SGD',

requiredCountry: 'SG',

authMethodValidation: {

'GROUPLIMIT': {

requiresApproval: true,

mandatoryLimitCheck: true,

riskLevel: 'HIGH'

},

'AFPTHENLIMIT': {

requiresApproval: false,

mandatoryLimitCheck: false,

riskLevel: 'MEDIUM'

},

'AFPONLY': {

requiresApproval: false,

mandatoryLimitCheck: false,

riskLevel: 'LOW'

}

}

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reference Data Details Structure

Complete Reference Data Response

interface ReferenceDataDetails {

acctSys: string; // VAM, MDZ, MEPS, FAST

acctGrp: string; // SGB, RETAIL, CORPORATE

acctId: string; // Original account ID

country: string; // SG (Singapore)

currencyCode: string; // SGD

authMethod: string; // GROUPLIMIT, AFPTHENLIMIT, AFPONLY

riskLevel: string; // LOW, MEDIUM, HIGH

limitProfile: string; // {SYSTEM}\_{GROUP}\_{METHOD}

requiresApproval: boolean; // true for GROUPLIMIT

additionalAttributes: {

evaluatedAt: string; // ISO timestamp

evaluationRules: string; // Rule set used

sourceSystem: string; // Service identifier

complianceLevel: string; // SINGAPORE\_ENHANCED

authenticationTier: string; // TIER\_1, TIER\_2, TIER\_3

processingPriority: string; // STANDARD, ENHANCED, PRIORITY

[key: string]: string; // Extensible for future attributes

};

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Configuration

Environment Variables

# gRPC Configuration

GRPC\_PORT=50060

SERVICE\_NAME=fast-referencedata-service

# Service Configuration

LOG\_LEVEL=info

ENVIRONMENT=development

COUNTRY=SG

DEFAULT\_CURRENCY=SGD

# Business Rules Configuration

AUTH\_METHOD\_TIMEOUT\_MS=3000

MAX\_RETRY\_ATTEMPTS=3

RETRY\_BACKOFF\_MS=1000

USE\_MOCK\_DATA=true

# Cache Configuration

CACHE\_TTL\_SECONDS=300

CACHE\_MAX\_SIZE=10000

ENABLE\_CACHE=true

# Compliance Configuration

COMPLIANCE\_LEVEL=SINGAPORE\_ENHANCED

REQUIRED\_CURRENCY=SGD

REQUIRED\_COUNTRY=SG

ENABLE\_STRICT\_VALIDATION=true

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Database Schema

**Note: Currently implemented with in-memory business rules. Future production implementation may require:**

Authentication Rules Table

CREATE TABLE auth\_method\_rules (

rule\_id SERIAL PRIMARY KEY,

rule\_name VARCHAR(100) NOT NULL,

account\_pattern VARCHAR(100), -- Regex pattern for account matching

account\_system VARCHAR(20), -- VAM, MDZ, MEPS, FAST

account\_group VARCHAR(20), -- SGB, RETAIL, CORPORATE

min\_length INTEGER,

max\_length INTEGER,

auth\_method VARCHAR(20) NOT NULL, -- GROUPLIMIT, AFPTHENLIMIT, AFPONLY

risk\_level VARCHAR(10) NOT NULL, -- LOW, MEDIUM, HIGH

requires\_approval BOOLEAN DEFAULT false,

priority INTEGER DEFAULT 100, -- Rule precedence (lower = higher priority)

is\_active BOOLEAN DEFAULT true,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_auth\_rules\_pattern ON auth\_method\_rules(account\_pattern);

CREATE INDEX idx\_auth\_rules\_system ON auth\_method\_rules(account\_system);

CREATE INDEX idx\_auth\_rules\_priority ON auth\_method\_rules(priority, is\_active);

Reference Data Cache Table

CREATE TABLE reference\_data\_cache (

cache\_key VARCHAR(255) PRIMARY KEY,

account\_id VARCHAR(50) NOT NULL,

account\_system VARCHAR(20) NOT NULL,

auth\_method VARCHAR(20) NOT NULL,

ref\_data\_details JSONB NOT NULL,

cached\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP NOT NULL,

hit\_count INTEGER DEFAULT 0

);

CREATE INDEX idx\_refdata\_cache\_expiry ON reference\_data\_cache(expires\_at);

CREATE INDEX idx\_refdata\_cache\_account ON reference\_data\_cache(account\_id, account\_system);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Service Integration

Integration with Inward Processor Service

// Called by fast-inwd-processor-service after account lookup

const authMethodRequest: AuthMethodRequest = {

messageId: "uuid",

puid: "G3I123456789",

acctSys: "VAM", // From account lookup response

acctGrp: "SGB", // From account lookup response

acctId: "999888777666", // Original account ID

country: "SG", // Market context

currencyCode: "SGD", // Market context

metadata: {

messageType: "PACS.008",

processingTime: Date.now().toString()

},

timestamp: Date.now()

};

const response = await referenceDataClient.LookupAuthMethod(authMethodRequest);

Response Integration

// Response used for downstream routing decisions

if (response.success) {

const authMethod = response.authMethod;

const requiresApproval = response.refDataDetails.requiresApproval;

// Integration with orchestrator service

if (authMethod === 'GROUPLIMIT') {

// Route to limit check service (fire & forget)

await sendToLimitCheckService(message);

}

if (authMethod === 'AFPTHENLIMIT') {

// Enhanced validation required

await performEnhancedValidation(message);

}

// Standard AFP processing for all methods

await sendToAntifraudPlatform(message);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error Handling

Error Scenarios and Responses

| Error Type | Error Code | Description | Recovery Action |

|------------|------------|-------------|-----------------|

**| Invalid Request | REFDATA\_INVALID\_REQUEST | Missing required fields | Return validation error |**

**| Business Rule Error | REFDATA\_RULE\_ERROR | Rule evaluation failure | Use default auth method |**

**| Service Unavailable | REFDATA\_SERVICE\_ERROR | Internal service error | Retry with exponential backoff |**

**| Unknown Account Pattern | REFDATA\_UNKNOWN\_PATTERN | No matching rules | Default to AFPONLY |**

Error Response Example

{

"messageId": "550e8400-e29b-41d4-a716-446655440000",

"puid": "G3I1234567890123",

"success": false,

"authMethod": "",

"errorMessage": "Required field 'acct\_id' is missing",

"errorCode": "REFDATA\_INVALID\_REQUEST",

"refDataDetails": null,

"processedAt": 1640995200000,

"lookupSource": "STUB"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Performance Characteristics

Current Performance

* \*\*Average Response Time\*\*: ~50ms
* \*\*Success Rate\*\*: 100% (rule-based deterministic)
* \*\*Throughput\*\*: 2000+ requests per second
* \*\*Memory Usage\*\*: Low (in-memory rule evaluation)

Caching Strategy

// In-memory cache with TTL

class ReferenceDataCache {

private cache = new Map<string, CacheEntry>();

generateCacheKey(request: AuthMethodRequest): string {

return `${request.acctSys}:${request.acctId}:${request.country}`;

}

get(key: string): CacheEntry | null {

const entry = this.cache.get(key);

if (entry && entry.expiresAt > Date.now()) {

entry.hitCount++;

return entry;

}

this.cache.delete(key);

return null;

}

set(key: string, value: any, ttlSeconds: number): void {

this.cache.set(key, {

value,

cachedAt: Date.now(),

expiresAt: Date.now() + (ttlSeconds \* 1000),

hitCount: 0

});

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Monitoring and Health Checks

Health Check Implementation

async healthCheck(): Promise<HealthCheckResponse> {

try {

// Test business rules engine

const testAuthMethod = this.determineAuthMethod('TEST123');

if (!testAuthMethod) {

return {

status: 'NOT\_SERVING',

message: 'Business rules engine failure',

timestamp: Date.now()

};

}

return {

status: 'SERVING',

message: 'Reference data service is healthy and ready to serve requests',

timestamp: Date.now(),

additionalInfo: {

activeRules: this.getActiveRulesCount(),

cacheSize: this.cache.size,

processingMode: this.useMockData ? 'MOCK' : 'PRODUCTION'

}

};

} catch (error) {

return {

status: 'NOT\_SERVING',

message: `Health check failed: ${error instanceof Error ? error.message : 'Unknown error'}`,

timestamp: Date.now()

};

}

}

Monitoring Metrics

// Prometheus metrics

const metrics = {

authMethodRequests: new Counter({

name: 'referencedata\_requests\_total',

help: 'Total authentication method requests',

labelNames: ['auth\_method', 'account\_system', 'risk\_level']

}),

authMethodDuration: new Histogram({

name: 'referencedata\_duration\_seconds',

help: 'Authentication method lookup duration',

buckets: [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]

}),

ruleEvaluations: new Counter({

name: 'referencedata\_rule\_evaluations\_total',

help: 'Business rule evaluations',

labelNames: ['rule\_type', 'result']

}),

cacheHitRate: new Gauge({

name: 'referencedata\_cache\_hit\_rate',

help: 'Cache hit rate percentage'

})

};

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Security Considerations

Data Security

* \*\*No Sensitive Data Storage\*\*: Only business rules and metadata
* \*\*Audit Logging\*\*: Log all authentication method determinations
* \*\*Input Validation\*\*: Strict validation of all request parameters
* \*\*Rate Limiting\*\*: Prevent abuse of service endpoints

Compliance and Governance

* \*\*Rule Auditability\*\*: Track all rule changes and applications
* \*\*Regulatory Compliance\*\*: Align with Singapore banking regulations
* \*\*Data Governance\*\*: Maintain data lineage for authentication decisions
* \*\*Change Management\*\*: Controlled updates to business rules

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Future Enhancements

Production Readiness

* \*\*Database Integration\*\*: Persistent storage for business rules
* \*\*Rule Management UI\*\*: Administrative interface for rule configuration
* \*\*A/B Testing\*\*: Support for rule experimentation
* \*\*Machine Learning\*\*: Dynamic auth method optimization

Advanced Features

* \*\*Multi-Market Support\*\*: Rules for different countries and currencies
* \*\*Real-time Rule Updates\*\*: Hot reload of business rules
* \*\*Advanced Analytics\*\*: Authentication method effectiveness analysis
* \*\*Integration APIs\*\*: REST endpoints for external rule management

Scalability Improvements

* \*\*Distributed Caching\*\*: Redis cluster for rule caching
* \*\*Rule Versioning\*\*: Support for multiple rule versions
* \*\*Performance Optimization\*\*: Sub-millisecond rule evaluation
* \*\*Load Balancing\*\*: Geographic distribution of rule evaluation