[TES-3] Enterprise-Wide BSS Api Integration and Middleware Orchestration

JIRA REF

TES-3 - Enterprise-Wide BSS Api Integration and Middleware Orchestration DONE

1. Document Control

Version	History	Author(s)	Date	Approvals	Change Log
1.0	Initial draft for middleware APIs design	Senior Backend Architect	2025-07-01	Architecture Guild	Created design document for new middleware APIs supporting BSS integration use cases

---

2. Overview

Purpose

This document defines the detailed design of new and modified middleware APIs to support the integration of the new Business Support System (BSS) module with CRM, Billing, SCM, and Inventory systems. It provides a single, authoritative source of order, subscription, and charging logic exposed via versioned REST APIs for all consuming channels.

Scope

In scope:

- Design of new orchestrated middleware APIs for order placement, order snapshot retrieval, subscription modification, and customer 360° context aggregation.

- Integration orchestration with backend systems (BSS, CRM, Billing, SCM, Inventory).

- Data mapping, orchestration flows, and OpenAPI specifications for middleware APIs.

Out of scope:

- UI/UX redesign beyond API wiring.

- Legacy batch job refactoring.

- Data warehouse migrations.

Audience

- Middleware development team

- Backend service teams (BSS, CRM, Billing, SCM, Inventory)

- QA and testing teams

- Architecture guild and project managers

Definitions & Acronyms

- BSS: Business Support System

- CRM: Customer Relationship Management

- SCM: Supply Chain Management

- DTO: Data Transfer Object

- SLA: Service Level Agreement

- P95: 95th percentile latency

- HPA: Horizontal Pod Autoscaler

---

3. Architecture Summary

Component Overview

- Portal UI: Calls middleware APIs, displays consolidated data.

- Enterprise Middleware: Gateway and orchestrator, exposes versioned REST APIs, enforces security and schema validation, aggregates backend data.

- BSS Module: New system managing orders, subscriptions, charging, and product catalog.

- CRM: Customer profile and verification.

- Billing: Payment processing and invoicing.

- SCM: Shipment and fulfillment tracking.

- Inventory: Stock availability and reservation.

Integration Points

- Middleware orchestrates calls to CRM, Inventory, BSS, Billing, and SCM APIs.

- Uses OpenAPI 3.0 compliant interfaces for backend integration.

Assumptions

- All backend systems expose stable OpenAPI 3.0 specs.

- Middleware uses OAuth 2.0 for authentication.

- Backend systems respond within SLA to meet overall latency targets.

---

4. Middleware API Inventory

API Name	Type	Ownership	Path	Linked Design Doc / OpenAPI Spec
Place Order	Composite	New	POST /orders	[Design Link]
Get Consolidated Order Snapshot	Composite	New	GET /orders/{orderId}	[Design Link]
Modify Subscription Plan	Simple	New	PATCH /subscriptions/{subId}	[Design Link]
Get 360° Customer Context	Composite	New	GET /customer/{customerId}/context	[Design Link]

---

5. API Detailed Design

5.1 Place Order (POST /orders)

5.1.1 Overview

Summary:

Allows portal users to place an order by submitting customerId and basket items. Middleware orchestrates verification, stock reservation, order creation, and payment charging.

API Owner: Middleware Team Lead

Purpose: Create an order and initiate payment via backend orchestration.

Authentication/Authorization: OAuth 2.0 token required with scope `order:create`.

Error Handling: Returns HTTP 400 for invalid payload, 404 for unknown customer, 409 for insufficient stock, 402 for payment failure, and 500 for internal errors.

5.1.2 Orchestration

API Type: Composite

Backend/Other APIs Called:

1. CRM: GET /crm/customers/{customerId} – validate customer existence and status.

2. Inventory: POST /inventory/reservations – reserve stock for each SKU.

3. BSS: POST /bss/orders – create order record.

4. Billing: POST /billing/charges – initiate payment charge.

Sequence:

a. Validate customer via CRM.

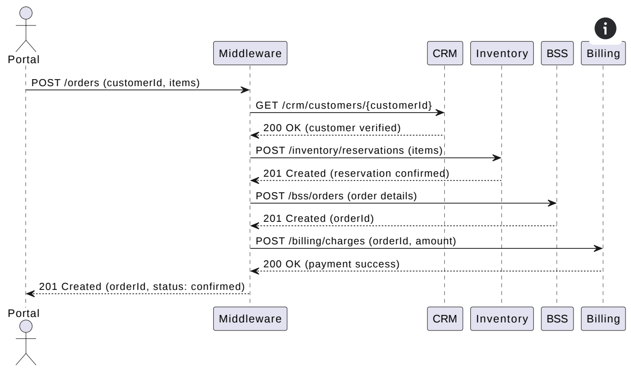
b. Reserve stock in Inventory.

c. Create order in BSS.

d. Charge payment in Billing.

e. Aggregate orderId and payment status for response.

UML Diagram:


Portal
Middleware
CRM
Inventory
BSS
Billing
POST /orders (customerId, items)
GET /crm/customers/{customerId}
200 OK (customer verified)
POST /inventory/reservations (items)
201 Created (reservation confirmed)
POST /bss/orders (order details)
201 Created (orderId)
POST /billing/charges (orderId, amount)
200 OK (payment success)
201 Created (orderId, status: confirmed)

Orchestration Steps:

a. Step 1: Validate customer existence and status in CRM.

b. Step 2: Reserve stock for requested SKUs in Inventory.

c. Step 3: Create order in BSS system.

d. Step 4: Initiate payment charge in Billing.

e. Step 5: Aggregate and respond with orderId and payment status.

5.1.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
POST /orders	customerId	GET /crm/customers/{id}	id	string	Used for customer validation
POST /orders	items	POST /inventory/reservations	sku	string	SKU of item to reserve
POST /orders	items	POST /inventory/reservations	quantity	integer	Quantity to reserve
POST /orders	customerId	POST /bss/orders	customerId	string	Order creation payload
POST /orders	items	POST /bss/orders	items[].sku	string	SKU list for order
POST /orders	items	POST /bss/orders	items[].quantity	integer	Quantity per SKU
POST /orders	orderId (from BSS response)	POST /billing/charges	orderId	string	Used to initiate payment charge
POST /orders	// TODO	POST /billing/charges	amount	number	Calculated total amount for charge

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute	Backend API Name (Path)	Response Attribute	Data Type	Notes

	(Northbound)		(Southbound)		
POST /orders	orderId	POST /bss/orders	orderId	string	Returned order identifier
POST /orders	status	POST /bss/orders	status	string	Order creation status
POST /orders	paymentStatus	POST /billing/charges	status	string	Payment charge status

---

5.1.4 Sample Request/Response

Sample Request:

```
json
{
  "customerId": "C12345",
  "items": [
    {
      "sku": "SKU123",
      "quantity": 2
    },
    {
      "sku": "SKU456",
      "quantity": 1
    }
  ]
}
```

Sample Response:

```
json
{
  "orderId": "ORD98765",
  "status": "confirmed",
  "paymentStatus": "success"
}
```

5.1.5 OpenAPI Specification (Middleware API)

```
yaml
openapi: 3.0.3
info:
  title: Place Order API
  version: 1.0.0
paths:
```

```yaml
  /orders:
    post:
      summary: Place an order with customer and basket details
      operationId: placeOrder
      security:
        - oauth2: [order:create]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - customerId
                - items
              properties:
                customerId:
                  type: string
                  example: "C12345"
                items:
                  type: array
                  items:
                    type: object
                    required:
                      - sku
                      - quantity
                    properties:
                      sku:
                        type: string
                        example: "SKU123"
                      quantity:
                        type: integer
                        example: 2
      responses:
        '201':
          description: Order created successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  orderId:
                    type: string
                    example: "ORD98765"
                  status:
                    type: string
                    example: "confirmed"
                  paymentStatus:
                    type: string
                    example: "success"
        '400':
          description: Invalid request payload
        '404':
          description: Customer not found
        '409':
          description: Insufficient stock
        '402':
          description: Payment failure
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            order:create: Place orders
```

5.2 Get Consolidated Order Snapshot (GET /orders/{orderId})

5.2.1 Overview

Summary:

Retrieve a unified snapshot of order details, billing invoice, and shipment status for a given orderId.

API Owner: Middleware Team Lead

Purpose: Provide consolidated order lifecycle information.

Authentication/Authorization: OAuth 2.0 token with scope `order:read`.

Error Handling: 404 if order not found; partial data handled gracefully.

5.2.2 Orchestration

API Type: Composite

Backend/Other APIs Called:

- BSS: GET /bss/orders/{orderId}

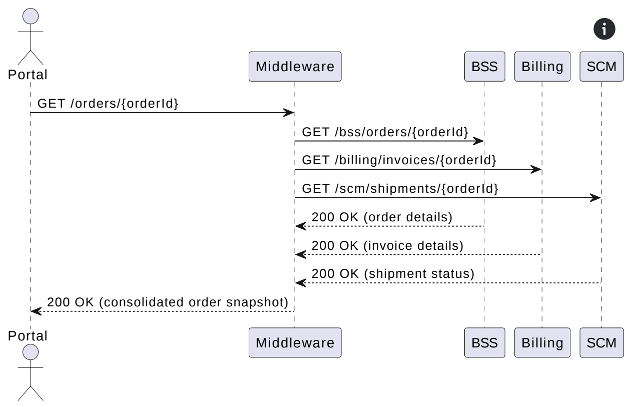
- Billing: GET /billing/invoices/{orderId}

- SCM: GET /scm/shipments/{orderId}

Sequence:

Parallel calls to BSS, Billing, SCM. Aggregate responses into unified DTO. Handle missing data gracefully.

UML Diagram:



Orchestration Steps:

a. Step 1: Retrieve order details from BSS.

b. Step 2: Retrieve invoice details from Billing.

c. Step 3: Retrieve shipment status from SCM.

d. Step 4: Aggregate and return consolidated response.

5.2.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
GET /orders/{orderId}	orderId	GET /bss/orders/{orderId}	id	string	Path parameter
GET /orders/{orderId}	orderId	GET /billing/invoices/{orderId}	orderId	string	Path parameter

GET /orders/{orderId}	orderId	GET /scm/shipments/{orderId}	orderId	string	Path parameter

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
GET /orders/{orderId}	orderDetails	GET /bss/orders/{orderId}	// TODO	object	Aggregated order detail fields
GET /orders/{orderId}	invoiceDetails	GET /billing/invoices/{orderId}	// TODO	object	Invoice information
GET /orders/{orderId}	shipmentStatus	GET /scm/shipments/{orderId}	status	string	Shipment status (e.g., "in transit")

5.2.4 Sample Request/Response

Sample Request:

```
GET /orders/ORD98765
```

Sample Response:

```
json
{
  "orderDetails": {
    // TODO - detailed order fields from BSS
  },
  "invoiceDetails": {
    // TODO - detailed invoice fields from Billing
  },
  "shipmentStatus": "in transit"
}
```

5.2.5 OpenAPI Specification (Middleware API)

```yaml
openapi: 3.0.3
info:
  title: Get Consolidated Order Snapshot API
  version: 1.0.0
paths:
  /orders/{orderId}:
    get:
      summary: Get consolidated order, billing, and shipment details
      operationId: getOrderSnapshot
      security:
        - oauth2: [order:read]
      parameters:
        - name: orderId
          in: path
          required: true
          schema:
            type: string
          example: "ORD98765"
      responses:
        '200':
          description: Consolidated order snapshot
          content:
            application/json:
              schema:
                type: object
                properties:
                  orderDetails:
                    type: object
                    description: Order details from BSS
                  invoiceDetails:
                    type: object
                    description: Invoice details from Billing
                  shipmentStatus:
                    type: string
                    description: Shipment status from SCM
                    example: "in transit"
        '404':
          description: Order not found
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            order:read: Read order information
```

5.3 Modify Subscription Plan (PATCH /subscriptions/{subId})

5.3.1 Overview

Summary:

Allows authorized users to upgrade or downgrade a subscription plan.

API Owner: Middleware Team Lead

Purpose: Update subscription plan in BSS and reflect changes in Billing.

Authentication/Authorization: OAuth 2.0 with scope `subscription:modify`.

Error Handling: 404 for invalid subscriptionId, 400 for invalid plan values.

5.3.2 Orchestration

API Type: Simple

Backend/Other APIs Called:

- BSS: PATCH /bss/subscriptions/{id} – update subscription plan.

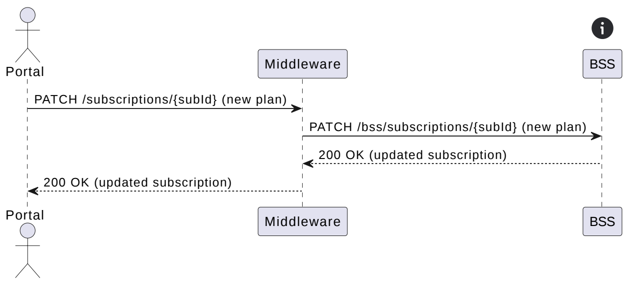
Sequence:

a. Validate plan details.

b. Update subscription plan in BSS.

c. Return updated subscription status.

UML Diagram:


Portal
Middleware
BSS
PATCH /subscriptions/{subId} (new plan)
PATCH /bss/subscriptions/{subId} (new plan)
200 OK (updated subscription)
200 OK (updated subscription)
Portal
Middleware
BSS

5.3.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
PATCH /subscriptions/{subId}	plan	PATCH /bss/subscriptions/{id}	plan	string	New subscription plan value

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
PATCH /subscriptions/{subId}	id	PATCH /bss/subscriptions/{id}	id	string	Subscription identifier
PATCH /subscriptions/{subId}	plan	PATCH /bss/subscriptions/{id}	plan	string	Updated plan
PATCH /subscriptions/{subId}	status	PATCH /bss/subscriptions/{id}	status	string	Subscription status

---

5.3.4 Sample Request/Response

Sample Request:

```
json
{
  "plan": "Premium"
}
```

Sample Response:

```
json
{
  "id": "SUB1122",
  "plan": "Premium",
  "status": "active"
}
```

5.3.5 OpenAPI Specification (Middleware API)

```
yaml
openapi: 3.0.3
info:
  title: Modify Subscription Plan API
  version: 1.0.0
paths:
  /subscriptions/{subId}:
    patch:
      summary: Modify subscription plan
      operationId: modifySubscriptionPlan
      security:
        - oauth2: [subscription:modify]
      parameters:
        - name: subId
          in: path
          required: true
          schema:
            type: string
          example: "SUB1122"
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - plan
              properties:
                plan:
                  type: string
                  example: "Premium"
      responses:
        '200':
          description: Subscription updated successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: string
                    example: "SUB1122"
                  plan:
                    type: string
                    example: "Premium"
                  status:
                    type: string
                    example: "active"
        '400':
          description: Invalid plan value
        '404':
          description: Subscription not found
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            subscription:modify: Modify subscription plans
```

5.4 Get 360° Customer Context (GET /customer/{customerId}/context)

5.4.1 Overview

Summary:

Retrieve a comprehensive view of customer profile, active subscriptions, outstanding balances, and recent shipments.

API Owner: Middleware Team Lead

Purpose: Provide support agents with unified customer context.

Authentication/Authorization: OAuth 2.0 with scope `customer:read`.

Error Handling: 404 if customer not found; partial data handled gracefully.

5.4.2 Orchestration

API Type: Composite

Backend/Other APIs Called:

- CRM: GET /crm/customers/{customerId}

- BSS: GET /bss/subscriptions/{customerId} (assumed)

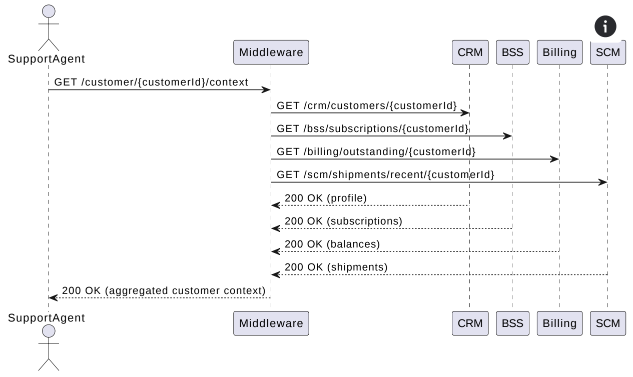
- Billing: GET /billing/outstanding/{customerId} (assumed)

- SCM: GET /scm/shipments/recent/{customerId} (assumed)

Sequence:

Parallel calls to backend systems; aggregate results into single response.

UML Diagram:


SupportAgent
Middleware
CRM
BSS
Billing
SCM
GET /customer/{customerId}/context
GET /crm/customers/{customerId}
GET /bss/subscriptions/{customerId}
GET /billing/outstanding/{customerId}
GET /scm/shipments/recent/{customerId}
200 OK (profile)
200 OK (subscriptions)
200 OK (balances)
200 OK (shipments)
200 OK (aggregated customer context)

5.4.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
GET /customer/{customerId}/context	customerId	GET /crm/customers/{id}	id	string	Customer profile lookup
GET /customer/{customerId}/context	customerId	GET /bss/subscriptions/{customerId}	customerId	string	Active subscriptions
GET /customer/{customerId}/context	customerId	GET /billing/outstanding/{customerId}	customerId	string	Outstanding balances
GET /customer/{customerId}/context	customerId	GET /scm/shipments/recent/{customerId}	customerId	string	Recent shipments

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
GET /customer/{customerId}/context	profile	GET /crm/customers/{id}	// TODO	object	Customer profile details
GET /customer/{customerId}/context	subscriptions	GET /bss/subscriptions/{customerId}	// TODO	array	List of active subscriptions
GET /customer/{customerId}/context	outstandingBalances	GET /billing/outstanding/{customerId}	// TODO	object	Outstanding balances info
GET /customer/{customerId}/context	recentShipments	GET /scm/shipments/recent/{customerId}	// TODO	array	Recent shipment details

---

5.4.4 Sample Request/Response

Sample Request:

`GET /customer/C12345/context`

Sample Response:

```
json
{
  "profile": {
    // TODO - detailed customer profile fields
  },
  "subscriptions": [
    // TODO - list of subscription objects
  ],
  "outstandingBalances": {
    // TODO - balance details
  },
  "recentShipments": [
    // TODO - shipment objects
  ]
}
```

5.4.5 OpenAPI Specification (Middleware API)

```yaml
openapi: 3.0.3
info:
  title: Get 360° Customer Context API
  version: 1.0.0
paths:
  /customer/{customerId}/context:
    get:
      summary: Get comprehensive customer context
      operationId: getCustomerContext
      security:
        - oauth2: [customer:read]
      parameters:
        - name: customerId
          in: path
          required: true
          schema:
            type: string
          example: "C12345"
      responses:
        '200':
          description: Aggregated customer context
          content:
            application/json:
              schema:
                type: object
                properties:
                  profile:
                    type: object
                    description: Customer profile data
                  subscriptions:
                    type: array
                    description: List of active subscriptions
                    items:
                      type: object
                  outstandingBalances:
                    type: object
                    description: Outstanding balance information
                  recentShipments:
                    type: array
                    description: Recent shipment details
                    items:
                      type: object
        '404':
          description: Customer not found
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            customer:read: Read customer context
```

---

6. Test Cases

Test Case Name	Description	Status Code	Expected Output
TC-01 Place Order - Happy Path	Valid customerId and items, sufficient stock, payment success	201	orderId returned, status "confirmed", paymentStatus "success"
TC-02 Place Order - Invalid Customer	Non-existent customerId	404	Error message "Customer not found"
TC-03 Place Order - Insufficient Stock	Stock reservation fails for one or more SKUs	409	Error message "Insufficient stock"
TC-04 Place Order - Payment Failure	Payment charge fails after order creation	402	Error message "Payment failure", compensating actions triggered
TC-05 Get Order Snapshot - Happy Path	Valid orderId returns consolidated data	200	Aggregated order, invoice, shipment info
TC-06 Get Order Snapshot - Order Not Found	Invalid orderId	404	Error message "Order not found"
TC-07 Modify Subscription - Happy Path	Valid subscriptionId and plan	200	Updated subscription details returned
TC-08 Modify Subscription - Invalid Subscription	Non-existent subscriptionId	404	Error message "Subscription not found"

TC-09 Modify Subscription - Invalid Plan	Plan value not allowed	400	Error message "Invalid plan value"
TC-10 Get Customer Context - Happy Path	Valid customerId returns full aggregated context	200	Full customer profile, subscriptions, balances, shipments
TC-11 Get Customer Context - Customer Not Found	Invalid customerId	404	Error message "Customer not found"

---

7. RAID Log

7.1 Risks

ID	Description	Impact	Mitigation	Owner	Status
R-01	Backend API latency exceeding SLA	High	Implement retries, circuit breakers	Middleware Team Lead	Open
R-02	Backend API schema changes post middleware release	Medium	Versioned APIs, contract testing	Architecture Guild	Open
R-03	Payment failure causing inconsistent order state	High	Compensating transactions, rollback	Middleware Team Lead	Open

7.2 Assumptions

ID	Assumption	Notes
A-01	All backend systems expose stable OpenAPI specs	Critical for data mapping and orchestration
A-02	OAuth 2.0 authentication is enforced consistently	Ensures secure API access

A-03	Backend systems respond within SLA to meet latency	Required to meet middleware P95 latency goals

7.3 Issues

ID	Description	Impact	Owner	Resolution Approach	Status
I-01	Partial data availability in order snapshot	Medium	Middleware Dev Lead	Graceful degradation handling	Open
I-02	Lack of downstream API for Billing outstanding balances	Medium	Middleware Team	Define interim mock or extension	Open

7.4 Dependencies

ID	Dependency	Notes
D-01	Backend teams to deliver OpenAPI specs on time	Delays affect middleware implementation
D-02	OAuth 2.0 authorization server availability	Required for secure token issuance
D-03	Portal UI to consume new middleware APIs	Coordination needed for rollout

---

End of Document

Backend OpenAPI Specification

```

### BSS API

openapi: 3.0.3
```

```
info:
  title: BSS API
version: 1.0.0
paths:
  orders:post
summary: Create OrderrequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:customer
    Id:
      type: stringitems
      type: arrayitems
      type: object
      properties:sku
    type: stringquantity
    type: integer
    example:customer
  Id: "C12345"items
sku: "SKU123"quantity
responses:'201'
description: Order Created
content:
  application/json:
    example:order
  Id: "ORD98765"status
get:
  summary: Get Subscriptionparameters
name: idin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Subscription Info
content:
  application/json:
    example:id
  plan: "Premium"status

---

### CRM API

openapi: 3.0.3
info:
  title: CRM API
version: 2.1.0
paths:/customers/{id}
get:
  summary: Get Customer Profileparameters
name: idin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Customer Details
content:
  application/json:
    example:id
  name: "John Doe"email
status: "verified"Billing API Spec

---

### Billing API

openapi: 3.0.3
info:
  title: Billing API
version: 1.2.0
paths:/charges
summary: Create Billing ChargerequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:order
    Id:
      type: stringamount
      type: number
    example:order
  Id: "ORD98765"amount
responses:
  '200':
description: Charge Initiated
content:
  application/json:
    example:charge
  Id: "CHG1234"status
get:
  summary: Fetch Invoice by Order IDparameters
name: orderIdin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Invoice Info
content:
  application/json:
    example:invoice
  Id: "INV4567"order
Id: "ORD98765"total

---

### SCM API

openapi: 3.0.3
info:
  title: SCM API
```

```
version: 1.0.0
paths:/shipments/{orderId}
get:
  summary: Get Shipment Infoparameters
name: orderIdin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Shipment Details
content:
  application/json:
    example:shipment
  Id: "SHIP7890"order
Id: "ORD98765"status

---

### Inventory API

openapi: 3.0.3
info:
  title: Inventory API
version: 1.1.0
paths:/reservations
summary: Reserve StockrequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:sku
    type: stringquantity
    type: integer
    example:sku
  quantity: 2
responses:'201'
description: Stock Reserved
content:
  application/json:
    example:reservation
  Id: "RSV001"status

```

Based on similar labels: BSS_Api

Related Pages

No related pages found for these labels.

JIRA REF
TES-3
1. Document Control

Version	History	Author(s)	Date	Approvals	Change Log
1.0	Initial draft for middleware APIs design	Senior Backend Architect	2025-07-01	Architecture Guild	Created design document for new middleware APIs supporting BSS integration use cases

2. Overview

Purpose

This document defines the detailed design of new and modified middleware APIs to support the integration of the new Business Support System (BSS) module with CRM, Billing, SCM, and Inventory systems. It provides a single, authoritative source of order, subscription, and charging logic exposed via versioned REST APIs for all consuming channels.

Scope

In scope:

- Design of new orchestrated middleware APIs for order placement, order snapshot retrieval, subscription modification, and customer 360° context aggregation.
- Integration orchestration with backend systems (BSS, CRM, Billing, SCM, Inventory).
- Data mapping, orchestration flows, and OpenAPI specifications for middleware APIs.

Out of scope:

- UI/UX redesign beyond API wiring.
- Legacy batch job refactoring.
- Data warehouse migrations.

Audience

- Middleware development team
- Backend service teams (BSS, CRM, Billing, SCM, Inventory)
- QA and testing teams
- Architecture guild and project managers

Definitions & Acronyms

- BSS: Business Support System
- CRM: Customer Relationship Management
- SCM: Supply Chain Management
- DTO: Data Transfer Object
- SLA: Service Level Agreement
- P95: 95th percentile latency
- HPA: Horizontal Pod Autoscaler

3. Architecture Summary

Component Overview

- Portal UI: Calls middleware APIs, displays consolidated data.
- Enterprise Middleware: Gateway and orchestrator, exposes versioned REST APIs, enforces security and schema validation, aggregates backend data.
- BSS Module: New system managing orders, subscriptions, charging, and product catalog.
- CRM: Customer profile and verification.
- Billing: Payment processing and invoicing.
- SCM: Shipment and fulfillment tracking.
- Inventory: Stock availability and reservation.

Integration Points

- Middleware orchestrates calls to CRM, Inventory, BSS, Billing, and SCM APIs.
- Uses OpenAPI 3.0 compliant interfaces for backend integration.

Assumptions

- All backend systems expose stable OpenAPI 3.0 specs.

- Middleware uses OAuth 2.0 for authentication.
- Backend systems respond within SLA to meet overall latency targets.

4. Middleware API Inventory

API Name	Type	Ownership	Path	Linked Design Doc / OpenAPI Spec
Place Order	Composite	New	POST /orders	[Design Link]
Get Consolidated Order Snapshot	Composite	New	GET /orders/{orderId}	[Design Link]
Modify Subscription Plan	Simple	New	PATCH /subscriptions/{subId}	[Design Link]
Get 360° Customer Context	Composite	New	GET /customer/{customerId}/context	[Design Link]

5. API Detailed Design

5.1 Place Order (POST /orders)
5.1.1 Overview
Summary:
Allows portal users to place an order by submitting customerId and basket items. Middleware orchestrates verification, stock reservation, order creation, and payment charging.
API Owner: Middleware Team Lead
Purpose: Create an order and initiate payment via backend orchestration.
Authentication/Authorization: OAuth 2.0 token required with scope order:create.
Error Handling: Returns HTTP 400 for invalid payload, 404 for unknown customer, 409 for insufficient stock, 402 for payment failure, and 500 for internal errors.
5.1.2 Orchestration
API Type: Composite
Backend/Other APIs Called:
1. CRM: GET /crm/customers/{customerId} – validate customer existence and status.
2. Inventory: POST /inventory/reservations – reserve stock for each SKU.
3. BSS: POST /bss/orders – create order record.
4. Billing: POST /billing/charges – initiate payment charge.
Sequence:
a. Validate customer via CRM.
b. Reserve stock in Inventory.
c. Create order in BSS.
d. Charge payment in Billing.
e. Aggregate orderId and payment status for response.
UML Diagram:

Orchestration Steps:
a. Step 1: Validate customer existence and status in CRM.
b. Step 2: Reserve stock for requested SKUs in Inventory.
c. Step 3: Create order in BSS system.
d. Step 4: Initiate payment charge in Billing.
e. Step 5: Aggregate and respond with orderId and payment status.

5.1.3 Data Mapping
1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
POST /orders	customerId	GET /crm/customers/{id}	id	string	Used for customer validation
POST /orders	items	POST /inventory/reservations	sku	string	SKU of item to reserve
POST /orders	items	POST /inventory/reservations	quantity	integer	Quantity to reserve
POST /orders	customerId	POST /bss/orders	customerId	string	Order creation payload
POST /orders	items	POST /bss/orders	items[].sku	string	SKU list for order
POST /orders	items	POST /bss/orders	items[].quantity	integer	Quantity per SKU
POST /orders	orderId (from BSS response)	POST /billing/charges	orderId	string	Used to initiate payment charge
POST /orders	// TODO	POST /billing/charges	amount	number	Calculated total amount for charge

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
POST /orders	orderId	POST /bss/orders	orderId	string	Returned order identifier
POST /orders	status	POST /bss/orders	status	string	Order creation status
POST /orders	paymentStatus	POST /billing/charges	status	string	Payment charge status

5.1.4 Sample Request/Response

Sample Request:

json
```
{
  "customerId": "C12345",
  "items": [
    {
      "sku": "SKU123",
      "quantity": 2
    },
    {
      "sku": "SKU456",
      "quantity": 1
    }
  ]
}
```

Sample Response:

json
```
{
  "orderId": "ORD98765",
  "status": "confirmed",
  "paymentStatus": "success"
}
```

5.1.5 OpenAPI Specification (Middleware API)

yaml
```
openapi: 3.0.3
info:
  title: Place Order API
```

```yaml
  version: 1.0.0
paths:
  /orders:
    post:
      summary: Place an order with customer and basket details
      operationId: placeOrder
      security:
        - oauth2: [order:create]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - customerId
                - items
              properties:
                customerId:
                  type: string
                  example: "C12345"
                items:
                  type: array
                  items:
                    type: object
                    required:
                      - sku
                      - quantity
                    properties:
                      sku:
                        type: string
                        example: "SKU123"
                      quantity:
                        type: integer
                        example: 2
      responses:
        '201':
          description: Order created successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  orderId:
                    type: string
                    example: "ORD98765"
                  status:
                    type: string
                    example: "confirmed"
                  paymentStatus:
                    type: string
                    example: "success"
        '400':
          description: Invalid request payload
        '404':
          description: Customer not found
        '409':
          description: Insufficient stock
        '402':
          description: Payment failure
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            order:create: Place orders
```

5.2 Get Consolidated Order Snapshot (GET /orders/{orderId})

5.2.1 Overview

Summary:

Retrieve a unified snapshot of order details, billing invoice, and shipment status for a given orderId.

API Owner: Middleware Team Lead

Purpose: Provide consolidated order lifecycle information.

Authentication/Authorization: OAuth 2.0 token with scope order:read.

Error Handling: 404 if order not found; partial data handled gracefully.

5.2.2 Orchestration

API Type: Composite

Backend/Other APIs Called:

- BSS: GET /bss/orders/{orderId}
- Billing: GET /billing/invoices/{orderId}
- SCM: GET /scm/shipments/{orderId}

Sequence:

Parallel calls to BSS, Billing, SCM. Aggregate responses into unified DTO. Handle missing data gracefully.

UML Diagram:

Orchestration Steps:

a. Step 1: Retrieve order details from BSS.

b. Step 2: Retrieve invoice details from Billing.

c. Step 3: Retrieve shipment status from SCM.

d. Step 4: Aggregate and return consolidated response.

5.2.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
GET /orders/{orderId}	orderId	GET /bss/orders/{orderId}	id	string	Path parameter
GET /orders/{orderId}	orderId	GET /billing/invoices/{orderId}	orderId	string	Path parameter
GET /orders/{orderId}	orderId	GET /scm/shipments/{orderId}	orderId	string	Path parameter

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
GET /orders/{orderId}	orderDetails	GET /bss/orders/{orderId}	// TODO	object	Aggregated order detail fields
GET /orders/{orderId}	invoiceDetails	GET /billing/invoices/{orderId}	// TODO	object	Invoice information
GET /orders/{orderId}	shipmentStatus	GET /scm/shipments/{orderId}	status	string	Shipment status (e.g., "in transit")

5.2.4 Sample Request/Response

Sample Request:

```
GET /orders/ORD98765
```

Sample Response:

json

```json
{
  "orderDetails": {
    // TODO - detailed order fields from BSS
  },
  "invoiceDetails": {
    // TODO - detailed invoice fields from Billing
  },
  "shipmentStatus": "in transit"
}
```

5.2.5 OpenAPI Specification (Middleware API)

yaml

```yaml
openapi: 3.0.3
info:
  title: Get Consolidated Order Snapshot API
  version: 1.0.0
paths:
  /orders/{orderId}:
    get:
      summary: Get consolidated order, billing, and shipment details
      operationId: getOrderSnapshot
      security:
        - oauth2: [order:read]
      parameters:
        - name: orderId
          in: path
          required: true
          schema:
```

```
        type: string
      example: "ORD98765"
   responses:
    '200':
     description: Consolidated order snapshot
     content:
      application/json:
       schema:
        type: object
        properties:
         orderDetails:
          type: object
          description: Order details from BSS
         invoiceDetails:
          type: object
          description: Invoice details from Billing
         shipmentStatus:
          type: string
          description: Shipment status from SCM
          example: "in transit"
    '404':
     description: Order not found
    '500':
     description: Internal server error
components:
 securitySchemes:
  oauth2:
   type: oauth2
   flows:
    clientCredentials:
     tokenUrl: https://auth.example.com/oauth2/token
     scopes:
      order:read: Read order information
```

5.3 Modify Subscription Plan (PATCH /subscriptions/{subId})

5.3.1 Overview

Summary:

Allows authorized users to upgrade or downgrade a subscription plan.

API Owner: Middleware Team Lead

Purpose: Update subscription plan in BSS and reflect changes in Billing.

Authentication/Authorization: OAuth 2.0 with scope subscription:modify.

Error Handling: 404 for invalid subscriptionId, 400 for invalid plan values.

5.3.2 Orchestration

API Type: Simple

Backend/Other APIs Called:

- BSS: PATCH /bss/subscriptions/{id} – update subscription plan.

Sequence:

a. Validate plan details.

b. Update subscription plan in BSS.

c. Return updated subscription status.

UML Diagram:

5.3.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
PATCH /subscriptions/{subId}	plan	PATCH /bss/subscriptions/{id}	plan	string	New subscription plan value

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
PATCH /subscriptions/{subId}	id	PATCH /bss/subscriptions/{id}	id	string	Subscription identifier
PATCH /subscriptions/{subId}	plan	PATCH /bss/subscriptions/{id}	plan	string	Updated plan
PATCH /subscriptions/{subId}	status	PATCH /bss/subscriptions/{id}	status	string	Subscription status

5.3.4 Sample Request/Response

Sample Request:

json
```
{
  "plan": "Premium"
}
```
Sample Response:

json
```
{
  "id": "SUB1122",
  "plan": "Premium",
  "status": "active"
}
```

5.3.5 OpenAPI Specification (Middleware API)

yaml
```
openapi: 3.0.3
info:
  title: Modify Subscription Plan API
  version: 1.0.0
paths:
  /subscriptions/{subId}:
    patch:
      summary: Modify subscription plan
      operationId: modifySubscriptionPlan
      security:
        - oauth2: [subscription:modify]
      parameters:
        - name: subId
```

```yaml
          in: path
          required: true
          schema:
            type: string
          example: "SUB1122"
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - plan
              properties:
                plan:
                  type: string
                  example: "Premium"
      responses:
        '200':
          description: Subscription updated successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: string
                    example: "SUB1122"
                  plan:
                    type: string
                    example: "Premium"
                  status:
                    type: string
                    example: "active"
        '400':
          description: Invalid plan value
        '404':
          description: Subscription not found
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            subscription:modify: Modify subscription plans
```

5.4 Get 360° Customer Context (GET /customer/{customerId}/context)

5.4.1 Overview

Summary:

Retrieve a comprehensive view of customer profile, active subscriptions, outstanding balances, and recent shipments.

API Owner: Middleware Team Lead

Purpose: Provide support agents with unified customer context.

Authentication/Authorization: OAuth 2.0 with scope customer:read.

Error Handling: 404 if customer not found; partial data handled gracefully.

5.4.2 Orchestration

API Type: Composite

Backend/Other APIs Called:

- CRM: GET /crm/customers/{customerId}
- BSS: GET /bss/subscriptions/{customerId} (assumed)
- Billing: GET /billing/outstanding/{customerId} (assumed)
- SCM: GET /scm/shipments/recent/{customerId} (assumed)

Sequence:

Parallel calls to backend systems; aggregate results into single response.

UML Diagram:

5.4.3 Data Mapping

1. Request Data Mapping

Middleware API Name (Path)	Request Attribute (Northbound)	Backend API Name (Path)	Request Attribute (Southbound)	Data Type	Notes
GET /customer/{customerId}/context	customerId	GET /crm/customers/{id}	id	string	Customer profile lookup
GET /customer/{customerId}/context	customerId	GET /bss/subscriptions/{customerId}	customerId	string	Active subscriptions
GET /customer/{customerId}/context	customerId	GET /billing/outstanding/{customerId}	customerId	string	Outstanding balances
GET /customer/{customerId}/context	customerId	GET /scm/shipments/recent/{customerId}	customerId	string	Recent shipments

2. Response Data Mapping

Middleware API Name (Path)	Response Attribute (Northbound)	Backend API Name (Path)	Response Attribute (Southbound)	Data Type	Notes
GET /customer/{customerId}/context	profile	GET /crm/customers/{id}	// TODO	object	Customer profile details
GET /customer/{customerId}/context	subscriptions	GET /bss/subscriptions/{customerId}	// TODO	array	List of active subscriptions
GET /customer/{customerId}/context	outstandingBalances	GET /billing/outstanding/{customerId}	// TODO	object	Outstanding balances info
GET /customer/{customerId}/context	recentShipments	GET /scm/shipments/recent/{customerId}	// TODO	array	Recent shipment details

5.4.4 Sample Request/Response

Sample Request:

```
GET /customer/C12345/context
```

Sample Response:

```json
{
  "profile": {
    // TODO - detailed customer profile fields
  },
  "subscriptions": [
    // TODO - list of subscription objects
  ],
  "outstandingBalances": {
    // TODO - balance details
  },
  "recentShipments": [
    // TODO - shipment objects
  ]
}
```

5.4.5 OpenAPI Specification (Middleware API)

```yaml
openapi: 3.0.3
info:
  title: Get 360° Customer Context API
  version: 1.0.0
paths:
  /customer/{customerId}/context:
    get:
      summary: Get comprehensive customer context
      operationId: getCustomerContext
      security:
        - oauth2: [customer:read]
      parameters:
        - name: customerId
          in: path
          required: true
          schema:
            type: string
          example: "C12345"
      responses:
        '200':
          description: Aggregated customer context
          content:
            application/json:
              schema:
                type: object
                properties:
                  profile:
                    type: object
                    description: Customer profile data
                  subscriptions:
                    type: array
                    description: List of active subscriptions
                    items:
                      type: object
                  outstandingBalances:
                    type: object
                    description: Outstanding balance information
                  recentShipments:
                    type: array
                    description: Recent shipment details
                    items:
                      type: object
        '404':
          description: Customer not found
        '500':
          description: Internal server error
components:
  securitySchemes:
    oauth2:
      type: oauth2
      flows:
        clientCredentials:
          tokenUrl: https://auth.example.com/oauth2/token
          scopes:
            customer:read: Read customer context
```

6. Test Cases

Test Case Name	Description	Status Code	Expected Output
TC-01 Place Order - Happy Path	Valid customerId and items, sufficient stock, payment success	201	orderId returned, status "confirmed", paymentStatus "success"
TC-02 Place Order - Invalid Customer	Non-existent customerId	404	Error message "Customer not found"
TC-03 Place Order - Insufficient Stock	Stock reservation fails for one or more SKUs	409	Error message "Insufficient stock"
TC-04 Place Order - Payment Failure	Payment charge fails after order creation	402	Error message "Payment failure", compensating actions triggered
TC-05 Get Order Snapshot - Happy Path	Valid orderId returns consolidated data	200	Aggregated order, invoice, shipment info
TC-06 Get Order Snapshot - Order Not Found	Invalid orderId	404	Error message "Order not found"
TC-07 Modify Subscription - Happy Path	Valid subscriptionId and plan	200	Updated subscription details returned
TC-08 Modify Subscription - Invalid Subscription	Non-existent subscriptionId	404	Error message "Subscription not found"
TC-09 Modify Subscription - Invalid Plan	Plan value not allowed	400	Error message "Invalid plan value"
TC-10 Get Customer Context - Happy Path	Valid customerId returns full aggregated context	200	Full customer profile, subscriptions, balances, shipments
TC-11 Get Customer Context - Customer Not Found	Invalid customerId	404	Error message "Customer not found"

7. RAID Log

7.1 Risks

ID	Description	Impact	Mitigation	Owner	Status
R-01	Backend API latency exceeding SLA	High	Implement retries, circuit breakers	Middleware Team Lead	Open
R-02	Backend API schema changes post middleware release	Medium	Versioned APIs, contract testing	Architecture Guild	Open
R-03	Payment failure causing inconsistent order state	High	Compensating transactions, rollback	Middleware Team Lead	Open

7.2 Assumptions

ID	Assumption	Notes
A-01	All backend systems expose stable OpenAPI specs	Critical for data mapping and orchestration
A-02	OAuth 2.0 authentication is enforced consistently	Ensures secure API access
A-03	Backend systems respond within SLA to meet latency	Required to meet middleware P95 latency goals

7.3 Issues

ID	Description	Impact	Owner	Resolution Approach	Status
I-01	Partial data availability in order snapshot	Medium	Middleware Dev Lead	Graceful degradation handling	Open
I-02	Lack of downstream API for Billing outstanding balances	Medium	Middleware Team	Define interim mock or extension	Open

7.4 Dependencies

ID	Dependency	Notes
D-01	Backend teams to deliver OpenAPI specs on time	Delays affect middleware implementation
D-02	OAuth 2.0 authorization server availability	Required for secure token issuance
D-03	Portal UI to consume new middleware APIs	Coordination needed for rollout

End of Document

Backend OpenAPI Specification

BSS API

```
openapi: 3.0.3
info:
  title: BSS API
version: 1.0.0
paths:
  orders:post
summary: Create OrderrequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:customer
    Id:
      type: stringitems
      type: arrayitems
      type: object
      properties:sku
    type: stringquantity
    type: integer
    example:customer
  Id: "C12345"items
sku: "SKU123"quantity
responses:'201'
description: Order Created
content:
  application/json:
    example:order
  Id: "ORD98765"status
get:
  summary: Get Subscriptionparameters
name: idin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Subscription Info
content:
  application/json:
    example:id
  plan: "Premium"status
```

CRM API

```
openapi: 3.0.3
info:
  title: CRM API
version: 2.1.0
paths:/customers/{id}
get:
  summary: Get Customer Profileparameters
name: idin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Customer Details
content:
  application/json:
    example:id
  name: "John Doe"email
status: "verified"Billing API Spec
```

Billing API

```
openapi: 3.0.3
info:
  title: Billing API
version: 1.2.0
paths:/charges
summary: Create Billing ChargerequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:order
    Id:
      type: stringamount
      type: number
    example:order
  Id: "ORD98765"amount
responses:
  '200':
description: Charge Initiated
content:
  application/json:
    example:charge
  Id: "CHG1234"status
get:
  summary: Fetch Invoice by Order IDparameters
name: orderIdin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Invoice Info
content:
  application/json:
    example:invoice
  Id: "INV4567"order
Id: "ORD98765"total
```

SCM API

```
openapi: 3.0.3
info:
  title: SCM API
version: 1.0.0
paths:/shipments/{orderId}
get:
  summary: Get Shipment Infoparameters
name: orderIdin
pathrequired: true
schema:
  type: string
  responses:
    '200':
  description: Shipment Details
content:
  application/json:
    example:shipment
  Id: "SHIP7890"order
Id: "ORD98765"status
```

Inventory API

```
openapi: 3.0.3
info:
  title: Inventory API
version: 1.1.0
paths:/reservations
summary: Reserve StockrequestBody
required: true
content:
  application/json:
    schema:
      type: object
      properties:sku
    type: stringquantity
    type: integer
    example:sku
  quantity: 2
responses:'201'
description: Stock Reserved
content:
  application/json:
    example:reservation
  Id: "RSV001"status
```

Based on similar labels: BSS_Api

Related Pages

No related pages found for these labels.

JIRA REF

☑ TES-106 - TES-3 | Enterprise-Wide BSS Api Integration and Middleware Orchestration | bss_api DONE