Integration Technical Specification – HRG ↔ Recombo

Version: 1.0
Prepared by: Architecture Team

1. Document Overview

This document provides the technical specification for the integration between HRG's core verification platform and Recombo's AI-driven verification engine. It outlines the architectural design, communication patterns, security considerations, data exchange formats, and phased implementation strategy required to enable automated employment (EMP) and education (EDU) verification workflows.

The document is intended for engineering teams, architects, product owners, and vendor integration partners involved in designing, developing, testing, and supporting this integration.

1.1 Executive Summary

HRG is a global leader in background screening solutions, serving large enterprise customers with complex compliance and verification needs. Over time, HRG's verification processes require significant manual interpretation, rule validation, and researcher involvement. These manual steps introduce operational delays, increase total cost of ownership (TCO), and limit the overall scalability and throughput of HRG's verification ecosystem.

As part of HRG's technology modernization and digital transformation strategy, the organization intends to integrate with **Recombo's AI-driven verification engine** to automate candidate data validation with greater speed, accuracy, and consistency. Recombo's AI models provide automated matching, discrepancy detection, error identification logic aligned with HRG's verification rules.

This integration initiative aims to establish a seamless, secure, API-driven asynchronous communication channel between HRG's core platform and Recombo's AI platform. By automating verification workflows at scale, HRG expects to significantly reduce manual researcher effort, improve operational turnaround time (TAT), enhance result accuracy, and deliver a modernized customer experience.

HRG offers multiple verification products including education, employment, criminal, and others, each of which will be integrated with Recombo's AI capabilities in a phased approach to progressively achieve the organization's broader modernization and automation goals.

1.2 Objectives

The primary objectives of the HRG ↔ Recombo Integration

1. Seamless System-to-System Integration

- Establish secure REST-based communication between HRG's Fulfillment Service and Recombo's API layer.
- Use standardized JSON-based sub-requests (e.g., EMP/EDU) as input for Recombo's automated verification engine.
- Ensure asynchronous communication to support scalable and non-blocking workflows.

2. Modernization & Automation

- Minimize manual verification activities by utilizing Recombo's AI-driven matching, validation, and discrepancy detection models.
- Increase decision accuracy and reduce human error by applying structured rule-based AI logic aligned with HRG verification guidelines.

3. Scalability & Speed

- Enable real-time or near real-time processing from candidate submission to automated verification outcome.
- Improve throughput by delegating high-volume validation tasks to Recombo's automated processing pipeline.
- Support future growth across multiple product lines with a scalable integration pattern.

1.3 Scope

This document focuses exclusively on the technical integration between the HRG Core Platform and the Recombo AI Verification Platform, delivered in a phased implementation model.

It does **not** cover broader HRG modernization initiatives or enterprise-level capabilities such as:

- API Gateway enablement to eliminate point-to-point system integrations
- OAuth2, JWT-RSA, or hybrid authentication patterns
- Vendor subscription frameworks or multi-vendor orchestration models

The scope of this document is limited to the design, implementation, validation, and rollout of the HRG ⇔Recombo integration for automated verification workflows, beginning with

foundational communication and progressively expanding into full production-grade feature support.

Phased Implementation Approach

The integration will be executed across multiple phases to ensure stable rollout, validated communication patterns, and controlled expansion of functionality. Each phase builds on the previous one to minimize risk and ensure system readiness.

Phase 1: Foundational Connectivity & Stub-Based Integration

Phase 1 establishes the baseline infrastructure, connectivity, and API-level communication between HRG and Recombo. This phase focuses on validating that both systems can exchange messages reliably, authenticate using JWT (HMAC), and handle request/response formats—even before real verification data is processed.

1. Stub-Driven Integration Simulation

- Recombo simulates sending sample Sub Request IDs to HRG's Fulfillment Service.
- HRG Fulfillment Service sends **stubbed** verification requests to Recombo.
- Recombo responds with simulated payloads to complete the "round-trip" workflow.
- Both teams validate:
    - Network reachability
    - API invocation behavior (GET/POST)
    - Contract alignment
    - Error-free communication

2. Establish Foundational APIs

HRG builds an initial Fulfillment microservice exposure:

- **POST** /submitSubRequest
- **GET** /retrieveResults

Recombo exposes aligned endpoints to support Phase-1 validation.

3. Validate Authentication & Security

- Confirm JWT (HMAC) signing, header structure, and validation logic.
- Ensure message integrity, signature validation, and replay-attack protection.
- Secret key stored locally (no Azure Key Vault or HSM in Phase 1).

4. Readiness for Real-Time Integration

Validate:

- Request/response schemas
- Error codes
- Timeout behavior
- Logging, tracing, and observability signals

This phase ensures the ecosystem is ready for **Phase 2**, where real EMP/EDU data will begin flowing.

Phase 2: Real-Time Sub-Request Integration (Employment First)

Phase 2 introduces real-time communication and replaces stub payloads with actual employment verification data.

High-level scopes include:

- Integration of **Message-Oriented Middleware (RabbitMQ)** to feed ready sub-requests to the Fulfillment service.
- Real-time submission of HRG sub-requests (starting with **Employment**) to Recombo.
- End-to-end processing, including AI-based decision and callback.
- Retry mechanisms for transient or vendor-side failures.
- Completion of the full fulfillment workflow.
- Scope limited to **single employment sub-requests** (no edge cases or compound profiles).

Phase 3: Expanded Product Support & Compound Scenario Handling

Phase 3 extends capabilities to support additional product lines and more complex request types.

High-level scopes include:

- Full support for **Education** and **Employment** sub-requests.
- Handling of **manual sub-request generation** where applicable.
- Support for **compound scenarios** (limited complexity).

Phase 4: Future Enhancements

The scope of Phase 4 remains **TBD** and may include:

- Async callbacks with event streaming
- Advanced error orchestration
- AI model feedback loop integration

(Details to be refined based on business adoption and outcomes from previous phases.)

2 High-Level Architecture (Initial)

The following diagram represents the Phase-1 integration architecture between the **HRG Fulfillment Microservice** and the **Recombo 360 Look AI Platform**. Phase-1 focuses on foundational connectivity, stub-driven simulation, API contract validation, JWT authentication, and end-to-end message flow readiness.

2.1 Architecture Overview

Phase-1 is designed to verify:

- Recombo Adapter trigger the stub based subRequstID
- HRG → Recombo outbound POST requests
- Recombo → HRG callback POST requests
- JWT (HMAC) generation and validation
- Request/response data structure correctness
- Network connectivity and firewall path validation
- Logging and observability of the full round-trip

No real candidate data is processed in this phase; all Recombo-side components (Adapter, SubRequestID trigger flow, Rest API service) operate in simulation mode.

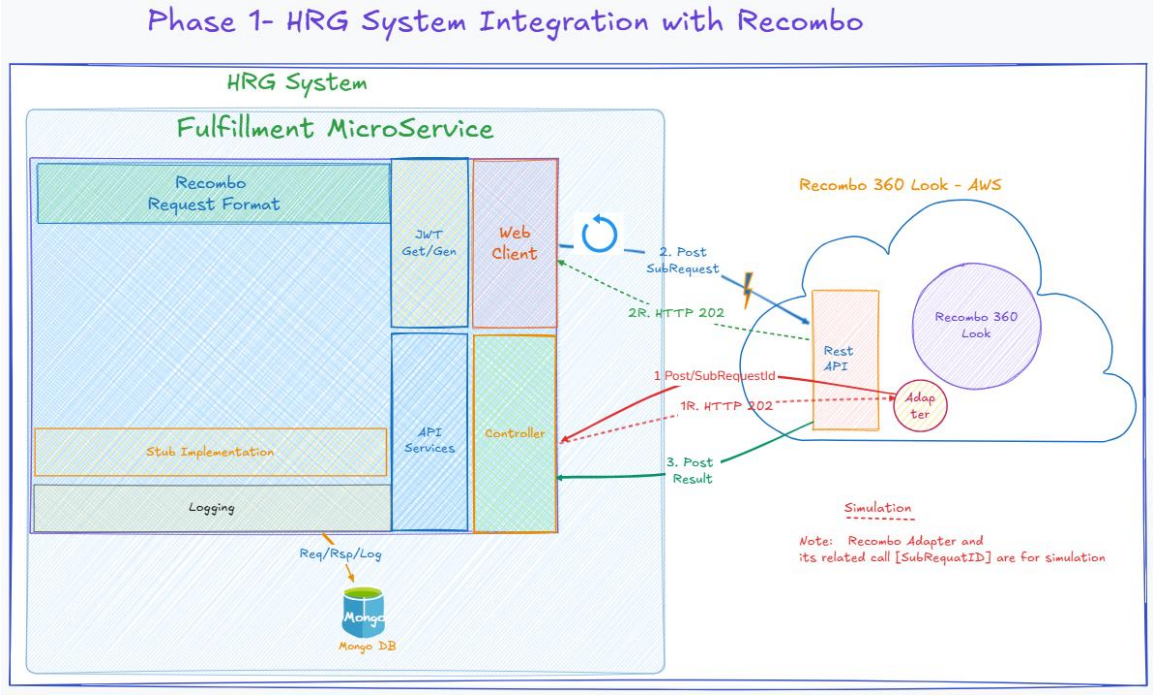


Figure 1: HRG Fulfillment System Integrate with Recombo

2.2 Components Overview

2.2.1 HRG Fulfillment Service (Phase-1)

A new microservice (Fulfillment Service) developed on **Spring Boot + JDK 21** that acts as the integration bridge between HRG and Recombo.

Responsibilities:

- Receive SubRequestID from Recombo (for simulation)
- Expose REST endpoints for Recombo to initiate verification requests
- Send stubbed verification requests to Recombo
- Handle simulated responses
- Maintain logs for request/response traces
- Validate JWT (HMAC) signed by Recombo
- Return API responses to Recombo in expected format

Endpoints exposed:

- POST / triggerSubRequest
- POST /result [finalResult]

2.2.2 Recombo AI Platform (Phase-1)

Recombo's integration layer exposes test endpoints to simulate the behavior of the AI verification engine.

Responsibilities:

- Send simulated SubRequestID to HRG Fulfillment Service
- Accept stubbed verification payload from HRG
- Return simulated verification results and outcomes
- Sign outgoing requests using JWT (HMAC)
- Validate HRG-signed JWT for incoming requests

Endpoints exposed:

- POST / submitSubRequest

2.2.3 JWT (HMAC) Based Authentication

A shared secret key (stored locally during Phase-1) is used to sign and validate JWT tokens between both systems.

Used for:

- Request integrity
- Authenticity assurance
- Replay attack protection through timestamp & nonce claims
  (No RSA public/private key infra in Phase-1)

2.2.4 6.2 JWT Token Structure

Both sides (HRG and Recombo) use:

- JWT (HMAC-SHA256)
- Shared secret key (pre-exchanged securely)

The sample JWT Header structure is given below:

Header:

```
{
 "alg": "HS256",
 "typ": "JWT"
}
```

Claims:

```
{
 "iss": "hrg-system",
 "aud": "recombo-api",
 "iat": 1731231231,
 "exp": 1731234531,
 "x-request-id": "HRG-REQ-12345",
 "scope": "fulfillment.submit"
}
```

2.3 High level Communication flow

Success case:

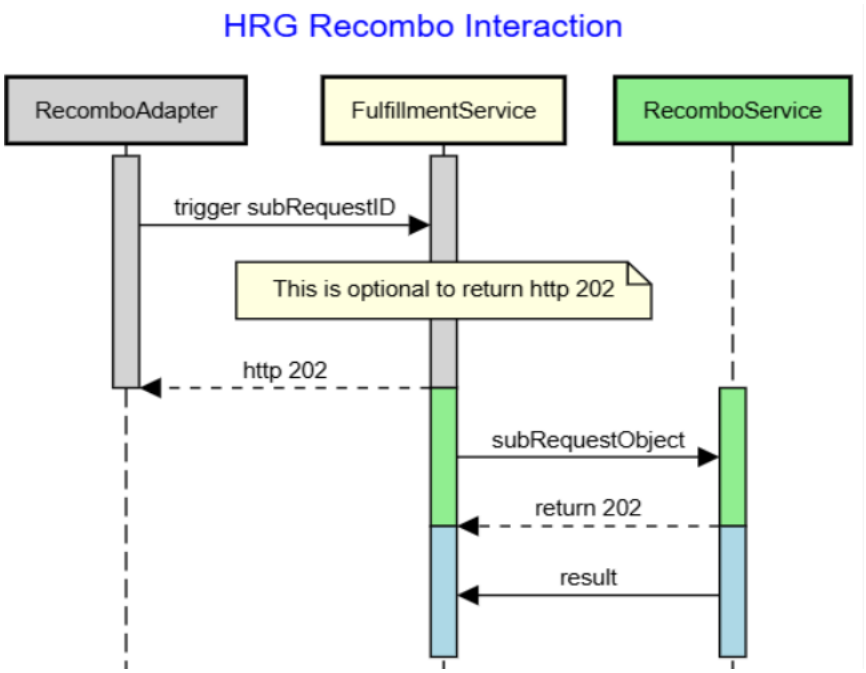


Figure 2: Sequence flow

Step	Sender	Receiver	API	Purpose
1	Recombo	HRG Fulfillment	POST /SubRequest	Start simulation with IDs
1R	HRG	Recombo	HTTP 202	Acknowledge.
2	HRG	Recombo	POST /submitSubRequest	Send detailed request JSON
2R	Recombo	HRG	HTTP 202	Acknowledgment
3	Recombo	HRG Fulfillment	POST /finalResult	Return simulated result

Retry/Failure case:

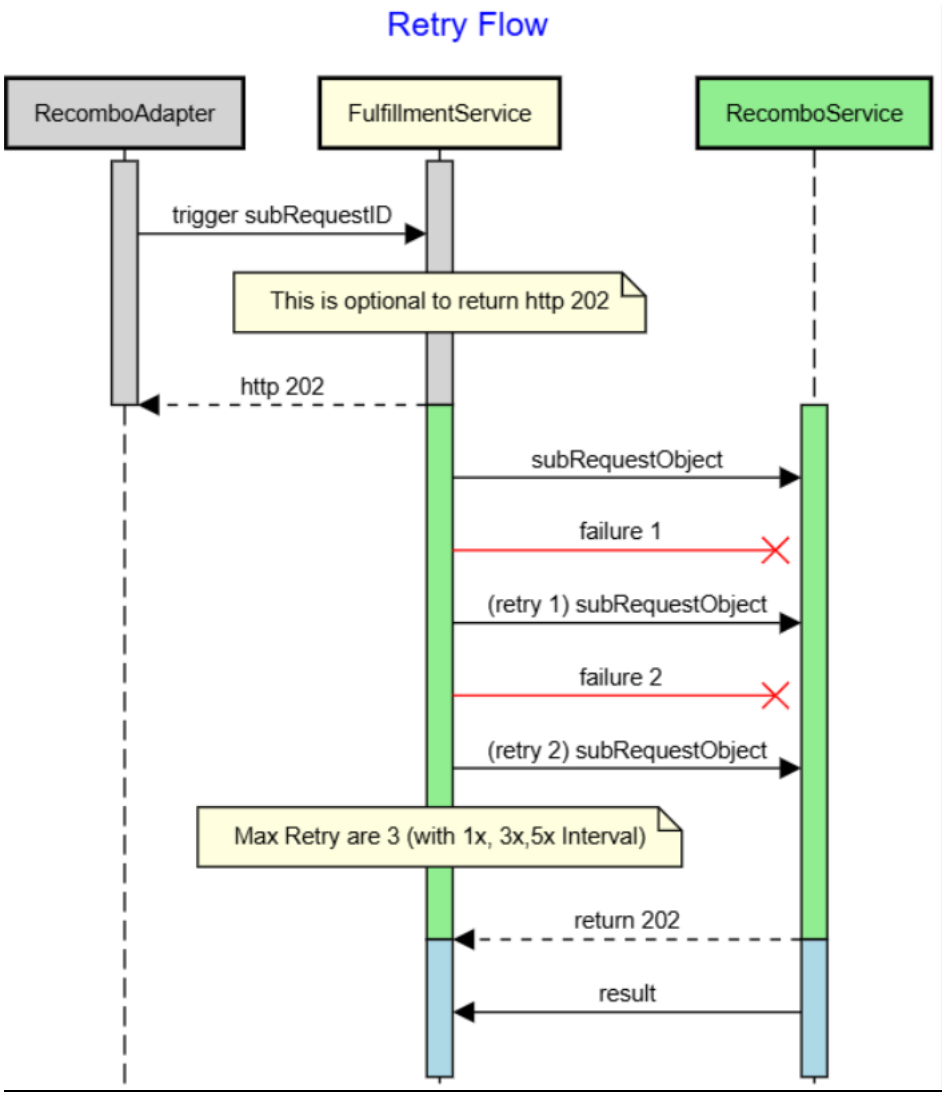


Figure 3: Retry sequence flow

2.4 API Endpoint Definitions (HRG ↔ Recombo)

This section defines the endpoints required for Phase-1 integration between the HRG Fulfillment Microservice and the Recombo AI Platform. These endpoints support the simulated round-trip verification flow used for validating connectivity, JWT authentication, message contracts, and callback mechanisms.

1. POST /triggerSubRequest

Caller: Recombo → HRG Fulfillment Microservice
Purpose:
Recombo notifies HRG that a new sub-request simulation should begin. HRG uses the provided identifiers to fetch/construct request details and initiate downstream processing.

Sample Request

https://<domainName>/v1/fulfillment/tasks/{taskid}/initiate {Note: <domainName> – Per environment}

BODY: <NONE>

Fulfillment Actions

- Validate incoming request
- Build Recombo request payload
- Generate JWT (HMAC)
- Call Recombo's /submitSubRequest endpoint with full payload [Please refer the Request format of the API]

Response

```
{
"status": "Accepted",
"message": "Sub-request triggered successfully."
 }
```

---

2. POST /submitSubRequest (Outgoing HRG → Recombo)

Caller: HRG Fulfillment Microservice → Recombo
Purpose:

HRG sends the **detailed request JSON**, which includes candidate and sub-request data, to Recombo for processing.

Note: This is an HRG-initiated outbound API call. It is *not* an API that HRG exposes publicly.

Request Body (example)


```
{
 "specversion": "1.0",
 "id": "hrg:hre:data-request:985702987",
 "source": "hrg:hre:data-request",
 "type": "EmploymentFulfillment",
 "datacontenttype": "application/json",
 "time": "2025-11-20T11:11:11Z",
 "dataschema": "schemas/EmploymentFulfillment.json",
 "data": {
  "hrgSource": {
   "product": {
    "sku": "VF-EM-IEMP-CORE-USA",
    "name": "Employment Verification Report"
   },
   "organization": {
    "name": "Tech Innovations Inc.",
    "location": {
     "city": "Berkeley",
     "region": "CA",
     "country": "US",
     "postalCode": "94720"
    },
    "tenure": {
     "start": "2018-03",
     "end": "",
     "current": true
    },
    "positionHistory": [
     {
      "verify": true,
      "title": "Senior Software Engineer",
      "tenure": {
       "start": "2018-03",
       "end": "",
       "current": true
      }
     }
    ]
   },
   "guidelines": [
    "Verify degree, major, and graduation date with registrar",
    "Confirm dates of attendance",
    "Rush processing requested - complete within 3 business days"
```

```
    ],
    "referenceObjects": [
     {
      "type": "order-item",
      "id": "hrg:hre:order-item:HE-020202-98765R-EM-001",
      "name": "similar institution"
     }
    ],
    "notes": [
     {
      "date": "",
      "originator": "",
      "note": ""
     }
    ]
   },
   "dataSource": {
    "name": "TALX",
    "type": "employment",
    "screenings": [
     {
      "orderServiceNo": "OR-122024-CZ896-EM-001",
      "employerOrgName": "Tech Innovations Inc.",
      "city": "Berkeley",
      "region": "CA",
      "country": "US",
      "positionHistory": {
       "title": "Software Engineer",
       "startDate": "2013-05-18",
       "endDate": "2017-05-22",
       "mostRecentHireDate": "2023-05-18",
       "employeeStatusCode": "8",
       "employeeStatus": "Inactive",
       "compensationInfo": {
        "currency": "USD",
        "rateOfPay": "",
        "averageHoursPerPayPeriod": "",
        "dateOfPayIncreaseLast": "notKnown",
        "amountOfPayIncreaseLast": "",
        "dateOfPayIncreaseNext": "notKnown",
        "amountOfPayIncreaseNext": ""
       }
      }
     },
     {
      "orderServiceNo": "OR-122024-CZ896-EM-002",
      "employerOrgName": "Tech Innovations Inc.",
      "city": "Berkeley",
      "region": "CA",
      "country": "US",
```

```
        "positionHistory": {
          "title": "Senior Software Engineer",
          "startDate": "2018-11-16",
          "endDate": "",
          "mostRecentHireDate": "2023-11-19",
          "employeeStatusCode": "8",
          "employeeStatus": "Active",
          "compensationInfo": {
            "currency": "USD",
            "rateOfPay": "",
            "averageHoursPerPayPeriod": "",
            "dateOfPayIncreaseLast": "notKnown",
            "amountOfPayIncreaseLast": "",
            "dateOfPayIncreaseNext": "notKnown",
            "amountOfPayIncreaseNext": ""
          }
        }
      }
    ]
  }
 }
}
```

POST /acknowledgement

Caller: Recombo → HRG Core (not Fulfillment MS)
Purpose:
Recombo sends an immediate **202-style acknowledgment** confirming that it has received the request and placed it into its internal processing flow.


```
{
"status": " Acknowledged ",
}
```


3. POST /finalResult

Caller: Recombo → HRG Fulfillment Microservice
Purpose:
After internal AI/ML processing, Recombo returns the **final verification decision** back to HRG.


```
{
 "specversion": "1.0",
 "id": "HE-010101-XXXXX-EM-001-CLASSIFICATION",
```

```
  "source": "hrg:hre:task",
  "type": "EmploymentFulfillmentTaskResult",
  "datacontenttype": "application/json",
  "dataschema": "tasks/EmploymentFulfillmentTaskResult",
  "data": {
   "decision": "VERIFIED_WITH_DISCREPANCY",
   "confidenceScore": 0.95,
   "decisionDescription": "Major discrepancy in title per guidelines",
   "resultData": {
    "employerOrgName": "Tech Innovations Inc.",
    "city": "Berkeley",
    "region": "CA",
    "country": "US",
    "positionHistory": {
     "title": "Software Engineer",
     "startDate": "2013-05-18",
     "endDate": "2017-05-22",
     "mostRecentHireDate": "2023-05-18",
     "employeeStatusCode": "8",
     "employeeStatus": "Inactive"
    }
   }
  }
}
```


POST/ acknowledgement -- Fulfillment MS Response


```
{
"status": " Acknowledged ",
}
```


3 High-Level Architecture (Final)

Once Phase-1 successfully validates communication, authentication, and message contracts, the integration advances to real-time operational mode. In this stage, HRG transitions from simulated request workflows to fully automated processing of actual verification data through Recombo's AI platform.

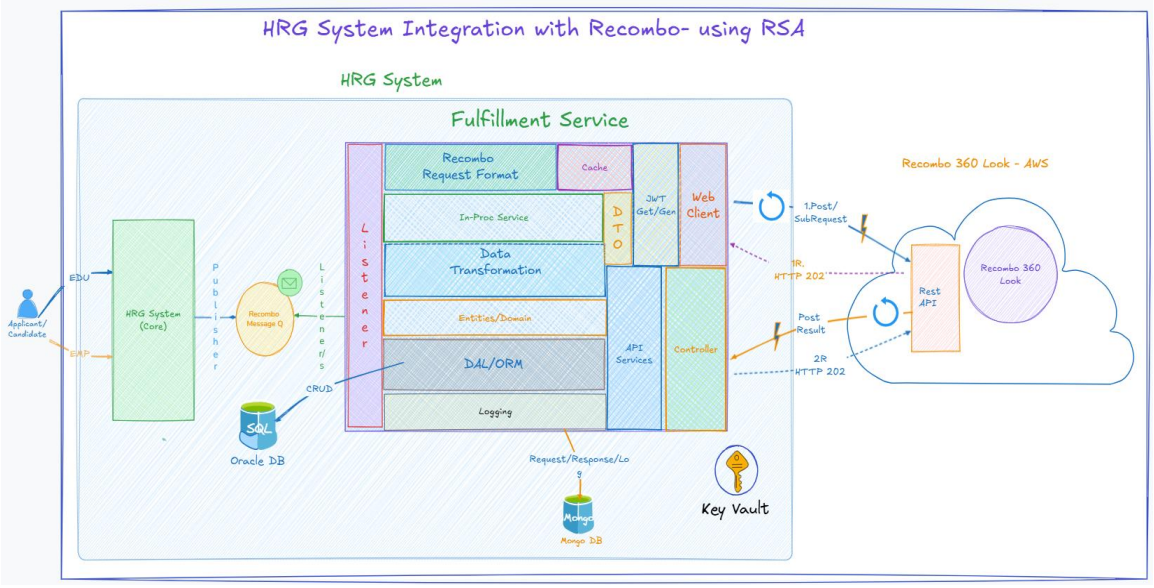


Figure 4: Realtime E2E Integration

3.1 Final Architecture Components

3.1.1 HRG Core System

- Accepts request from Applicant/Candidate or Integrated vendors
- Validate and store the candidate details in DB and prepare sub request based on rule. Invoke respective service like NSCH/TALX/Experian to get the candidate education / Employment details
- Publishes request messages to internal communication channel. (Rabbit MQ)

3.1.2 Fulfillment Service (Microservice)

Component	Purpose
Listener	Listens to messages (sub-request events) from HRG Core
In-Process Service	Internal processing coordination
Data Transformation	Converts HRG internal data → Recombo Format
DTO Layer	Holds structured request/response models
JWT Generation (HSAC)	Creates signed tokens using secrete key from Key Vault
Web Client	Invokes Recombo REST API asynchronously
Caching Layer	Stores prior requests, lookup values
API Services & Controller	Handles final callback responses
DAL/ORM	DB interaction with Oracle (Store and retrieve)
Logging	Tracks full workflow for auditing and traceability (both request and response formats)

Component	Purpose

Key Vault

- Stores secrete Key
- Accessed during JWT signing.

MongoDB

- Stores request/response logs.
- Retains traceability for operational analysis.

3.1.3 Recombo Platform (AWS)

- Receives sub-request details from HRG.
- Executes AI-driven validation.
- Returns result decisions via callback API.

3.1.4 E2E Flow

Below is the component level sequence flow of real time Fulfillment process.

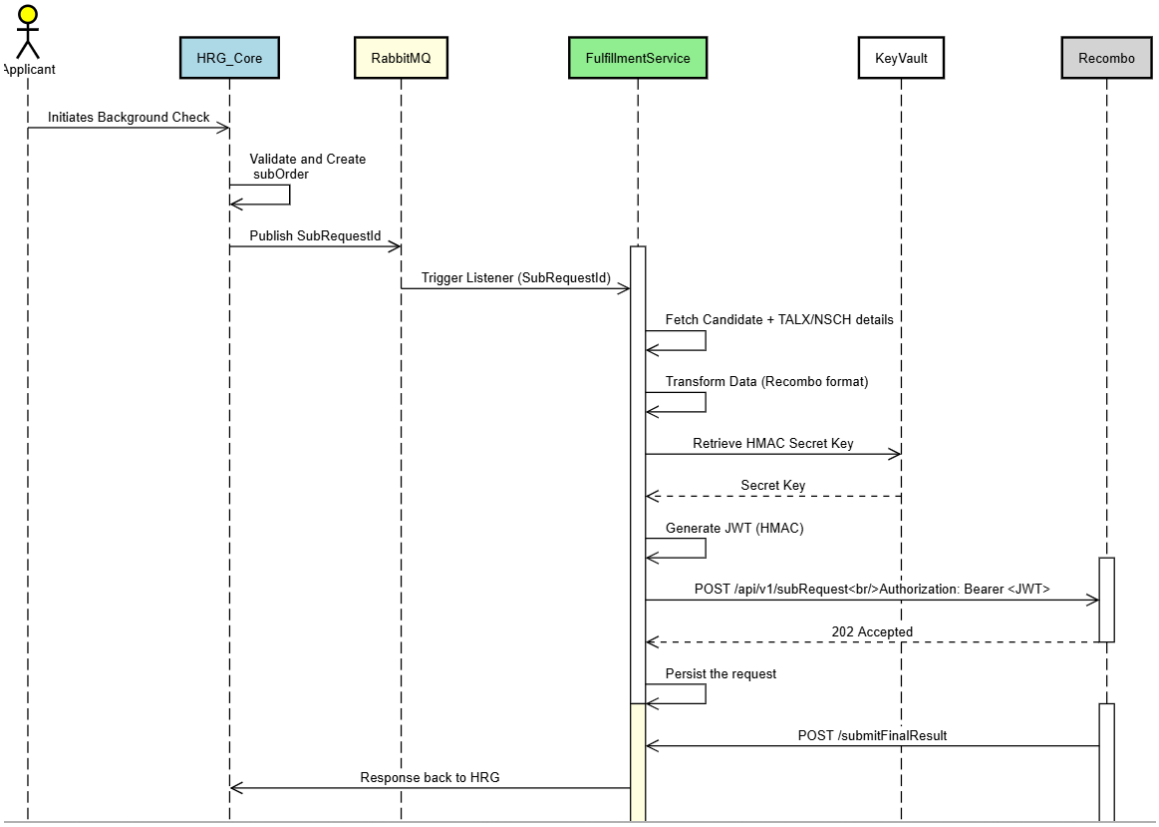


Figure 5: E2E Sequence flow

3.2 API Versioning Strategy

To ensure backward compatibility, maintain stability, and allow incremental enhancements without disrupting existing integrations, both HRG and Recombo systems will adopt a version-controlled API strategy.

Versioning Approach

- **API version will be included in the URL path.**
  Example:
  - /v1/subRequest
  - /v1/result
- **Version format**: v<major>[.<minor>]
  - Major versions (v1, v2, etc.) indicate breaking changes.
  - Minor versions (v1.1, v1.2) are optional and applicable only if required for non-breaking additions.
- **Current planned version:** v1
  All endpoints in **Phase 1** and **Phase 2** will use v1. Any enhancements identified post go-live will be addressed as v2.

Endpoint Versioning Example

Purpose	Endpoint
Send sub-request	POST /v1/subRequest
Final result callback from Recombo -> HRG	POST /v1/result

Version Deprecation Policy

- Both systems must support only **one active major version at a time**.
- Some **X-month deprecation window** will be provided before discontinuing older major versions.
- Deprecation notice will be shared:
  - In **API response headers** (Deprecation: true, Sunset-Date: <date>),
  - Via **technical change notification**,
  - Via **integration review meeting**.

API Version Validation

- Version mismatch must be detected during request validation.
- If unsupported version:

```
HTTP 400 Bad Request

{
  "errorCode": "UNSUPPORTED_API_VERSION",
  "message": "API version v1 is no longer supported. Please use v2."
}
```

3.3 Environment Requirements

To ensure a stable and secure integration between **HRG Fulfillment Service** and **Recombo AI Platform**, the environment readiness parameters below should be satisfied before moving to the next phase.

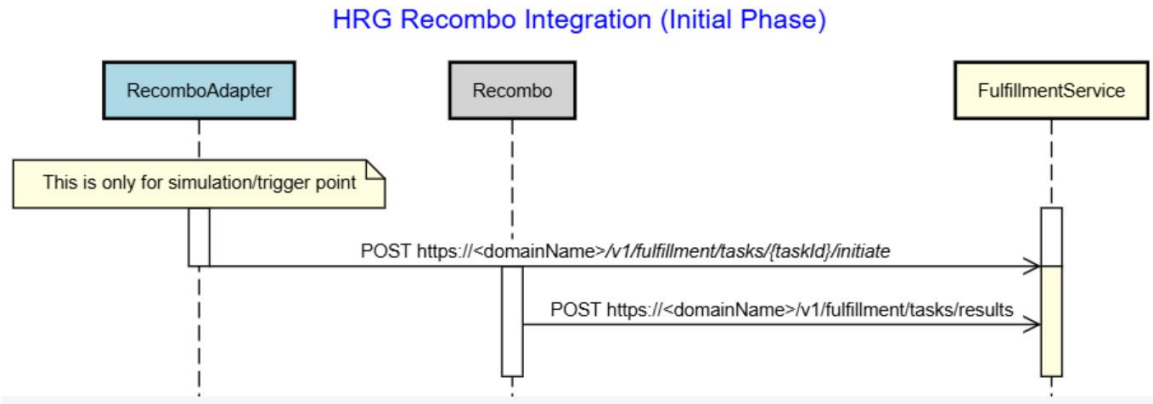
3.3.1 Required Environments

Environment	Purpose
DEV	Developer validation, internal unit testing, stub verification
TEST /QA	Integration testing with controlled test data
STAGE /PRE-PROD	Performance testing, security validation, and near-live simulation
PROD	Real-time execution of sub-request validations

3.3.2 Environment Readiness Checklist

Criteria	DEV	TEST	STAGE	PROD
E2E API Connectivity	✓	✓	✓	✓
JWT token validation	✓	✓	✓	✓
Network /VPN Setup	✓	✓	✓	✓
Error handling/retry testing	X	✓	✓	✓
Performance / Load testing	X	X	✓	✓
Alerts / Monitoring setup	X	Optional	Mostly Yes	Must
Security scan Test	X	X	✓	✓
Hight Availability	X	X	✓	✓

4 Request & Response



Recombo Adapter Initiate: **POST**
https://<domainName>/v1/fulfillment/tasks/{taskid}/initiate {Note: <domainName> – Per environment}

BODY: <NONE>

Recombo Final Response/Result: **POST**
https://<domainName>/v1/fulfillment/tasks/results

The sample BODY [When HTTP 200]:


```
{
  "specversion": "1.0",
  "id": "HE-010101-XXXXX-EM-001-CLASSIFICATION",
  "source": "hrg:hre:task",
  "type": "EmploymentFulfillmentResult",
  "datacontenttype": "application/json",
  "dataschema": "schemas/EmploymentFulfillmentResult.json",
  "data": {
    "decision": "VERIFIED_WITH_DISCREPANCY",
    "confidenceScore": 0.95,
    "decisionDescription": "Major discrepancy in title per guidelines",
    "resultData": {
      "employerOrgName": "Tech Innovations Inc.",
      "city": "Berkeley",
      "region": "CA",
      "country": "US",
      "positionHistory": {
        "title": "Software Engineer",
        "startDate": "2013-05-18",
        "endDate": "2017-05-22",
        "mostRecentHireDate": "2023-05-18",
        "employeeStatusCode": "8",
        "employeeStatus": "Inactive"
      }
    }
  }
}
```

Other HTTP Statues and formats are given below:

PENDING Status [HTTP 202]


```
{
 "specversion": "1.0",
 "id": "7a8f3c9e-2b44-4c11-b9e3-2f9a6d1a89d2",
 "source": "hrg:hre:fulfillment-api",
 "type": "Task.Accepted",
 "datacontenttype": "application/json",
 "time": "2025-11-20T10:15:30Z",
 "data": {
  "taskId": "HE-010101-9876",
  "status": "PENDING",
  "submittedAt": "2025-11-20T10:1Z",
  "expectedCompletion": "2025-11-20T10:17:00Z",
  "message": "Task request accepted for processing",
 }
}
```


When Authentication Failed [HTTP 401]


```
{
 "specversion": "1.0",
 "id": " {requestId}",
 "source": "hrg:hre:fulfillment",
 "type": "Error.Authentication",
 "datacontenttype": "application/json",
 "time": "2025-11-20T011:11:11Z ",
 "data": {
  "status": 401,
  "error": "Unauthorized",
  "message": "Authentication token is missing or invalid",
  "errorCode": "AUTH001",
  "reason": "INVALID_OR_EXPIRED_TOKEN"
 }
}
```


When Bad response from Results. [HTTP 400]


```
{
 "specversion": "1.0",
 "id": "" {requestId}",",
 "source": "hrg:hre:task",
 "type": "Error.Validation",
```

```
  "datacontenttype": "application/json",
  "time": "2025-11-20T011:11:11Z",
  "data": {
    "status": 400,
    "error": "Bad Request",
    "errorCode": "REQ001",
    "message": "Input validation failed",
    "details": [
      {"field": "decision", "message": "Decision is required" }
    ]
  }
}
```


5 Assumptions & Constraints

6 Appendix

Attached the Sample Json request and response formats of both education and Employment with single and compound requests.

RequestResponse.zip

Find the attached RequestResponse.zip contains

- edu_compound_request_v1.0.json
- emp_compound_request_v1.0.json
- edu_single_request_v1.0.json
- emp_single_request_v1.0.json
- edu_result_v1.0.json
- emp_result_v1.0.json

For schemas

- /schemas/EmploymentFulfillmentResult_v1.0.json
- /schemas/EducationFulfillmentResult_v1.0.json
- /schemas/EmploymentFulfillment_v1.0.json
- /schemas/EducationFulfillment_v1.0.json