Amazon-UPS Communication Protocol

1. Introduction

This document is protocol between the Mini-Amazon and Mini-UPS services as required for the ERSS project.

2. Communication Architecture

2.1 General Architecture

The communication between Amazon and UPS follows a REST-based architecture with JSON message payloads. Each service must expose HTTP endpoints for receiving messages from the other.

2.2 Message Format

All messages exchanged between Amazon and UPS shall be formatted as JSON and transmitted over HTTP. Each message must include:

- `action`: String identifying the message type
- `timestamp`: Time when the message was sent
- `message_id`: Unique UUID v4 identifier for the message

```
{
  "action": "string",
  "timestamp": "string",
  "message_id": "string (UUID)",
  ...other fields
}
```

2.3 Response Format

All responses must include:

- `action`: String identifying the response type (typically request action + "_response")
- `timestamp`: Time when the response was sent
- `message_id`: Unique UUID v4 identifier for the response
- `in_response_to`: The message_id of the request being responded to
- `status`: "success" or "error"
- `message`: Human-readable description, especially for errors

```
{
  "action": "string",
  "timestamp": "string ()",
  "message_id": "string (UUID)",
  "in_response_to": "string (UUID)",
  "status": "success | error",
  "message": "string",
  ...other fields specific to the response
}
```

2.4 Authentication

Requests between Amazon and UPS should include an authentication token in the HTTP Authorization header using the Bearer scheme:

```
Authorization: Bearer <token>
```

The token generation and distribution mechanism shall be coordinated separately between the Amazon and UPS teams.

3. Endpoints

3.1 Required Endpoints

Both services should expose the following HTTP endpoints:

- Amazon: `/api/ups` - Endpoint for receiving messages from UPS
- UPS: `/api/amazon` - Endpoint for receiving messages from Amazon

3.2 HTTP Methods

All requests must use the HTTP POST method.

3.3 Status Codes

Services must use appropriate HTTP status codes:

- 200 OK: Message processed successfully
- 400 Bad Request: Invalid message format
- 401 Unauthorized: Authentication failure
- 404 Not Found: Resource not found
- 500 Internal Server Error: Unexpected error

4. Package Creation and Pickup

4.1 Request Pickup (Amazon → UPS)

When Amazon needs to ship a package, it must send a pickup request to UPS.

```
{
  "action": "request_pickup",
  "warehouse_id": "integer",
  "user_id": "integer (optional)",
  "destination_x": "integer",
  "destination_y": "integer",
  "description": "string (optional)",
  "items": [
    {
      "name": "string",
      "description": "string (optional)",
      "quantity": "integer"
    },
    ...
  ]
}
```

UPS must respond with:

```
{
  "action": "pickup_response",
  "in_response_to": "",
  "status": "success | error",
  "tracking_number": "string (assigned tracking number)",
  "message": "string (description, especially for errors)"
}
```

4.2 Package Ready (Amazon → UPS)

When a package is packed and ready for pickup, Amazon MUST notify UPS.

```
{
  "action": "package_ready",
  "package_id": "string (tracking number)"
}
```
UPS must respond with:
```
{
  "action": "package_ready_response",
  "message_id": "",
  "in_response_to": "",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```
4.3 Truck Arrived (UPS → Amazon)
When a UPS truck arrives at a warehouse for pickup, UPS must notify Amazon.
```
{
  "action": "truck_arrived",
  "truck_id": "integer",
  "warehouse_id": "integer"
}
```
Amazon must respond with:
```
{
  "action": "truck_arrived_response",
  "in_response_to": "",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```
5. Package Loading
5.1 start Loading (Amazon → UPS)
When Amazon wants to load a package onto a truck, it MUST send:
```
{
  "action": "loading_package",
  "package_id": "string (tracking number)",
  "truck_id": "integer",
  "warehouse_id": "integer"
}
```
UPS must respond with:
```
{
  "action": "loading_package_response",
  "message_id": "",
  "in_response_to": "",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```
6. Delivery

6.1 Package Loaded (Amazon → UPS)
After a package is loaded, Amazon must notify UPS:
```
{
  "action": "package_loaded",
```

```
  "package_id": "string (tracking number)",
  "truck_id": "integer"
}
```
UPS must respond with:
```
{
  "action": "delivery_started",
  "in_response_to": "",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```

6.2 Package Delivered (UPS → Amazon)
When a package is delivered, UPS must notify Amazon:
```
{
  "action": "package_delivered",
  "package_id": "string (tracking number)",
  "truck_id": "integer",
  "delivery_x": "integer",
  "delivery_y": "integer"
}
```
Amazon respond with:
```
{
  "action": "package_delivered_response",
  "in_response_to": "ups",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```

7. Status Queries
7.1 Package Status Query (Amazon → UPS or UPS → Amazon)
Either service may query the status of a package:
```
{
  "action": "query_status",
   "package_id": "string (tracking number)"
}
```
The receiving service respond with:
```
{
  "action": "query_status_response",
  "in_response_to": "",
  "status": "success | error",
  "package_status": "created | waiting_for_pickup | pickup_assigned |
ready_for_pickup | pickup_complete | loading | loaded | out_for_delivery |
delivered",
  "truck_id": "integer (if assigned)",
  "truck_status": "idle | traveling | arrive_warehouse | loading | delivering
(if truck assigned)",
  "truck_location": {
    "x": "integer",
    "y": "integer"
  },
  "message": "string (description, especially for errors)"
}
```

```
Amazon response with:
{
      "truck_id":5,
      "package_status":""(string)
      "Action" : (string)
      "Status" :
      "In_response_to" :
}
```

8. Package Redirection

8.1 Redirect Package (UPS → Amazon)

When a user requests a package redirection through UPS, UPS must notify Amazon:

```
{
  "action": "redirect_package",
  "new_destination_x": "integer",
  "new_destination_y": "integer",
  "user_id": "integer"
}
```

Amazon must respond with:

```
{
  "action": "redirect_package_response",
  "in_response_to": "ups",
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```

9. Error Handling.

9.1 Error Types

- `validation_error`: Input validation failed
- `not_found`: Requested resource not found
- `conflict`: Business rule violation
- `unauthorized`: Authentication or authorization failure
- `internal_error`: Unexpected server error

10. World ID Coordination

10.1 World ID Sharing

To ensure both services connect to the same world in the world simulator, UPS must share the world ID with Amazon upon creation:

```
{
  "action": "world_created",
  "world_id": "integer"
}
```

Amazon must respond with:

```
{
  "action": "world_created_response",
  "in_response_to": "",
```

```
  "status": "success | error",
  "message": "string (description, especially for errors)"
}
```

11. Security

11.1 Authentication

Authentication must use either:

- API keys in HTTP headers
- JWT tokens
- Mutual TLS authentication

11.2 Input Validation

All services must validate all input data according to the following rules:

- `package_id`: Alphanumeric string between 8 and 20 characters
- `truck_id`: Positive integer between 1 and 1000
- `warehouse_id`: Positive integer between 1 and 100
- `x` and `y` coordinates: Integers between 0 and 100
- `user_id`: Positive integer

12. Detailed JSON Schema Definitions

12.1 Common Properties

```
{
  "type": "object",
  "required": ["action"],
  "properties": {
    "action": {
      "type": "string",
      "description": "The action type of the message"
    },
      "message_id": {
      "type": "string",
      "format": "uuid",
      "description": "Unique UUID v4 identifier for the message"
    }
  }
}
```

12.2 Response Properties

```
{
  "type": "object",
  "required": ["action","in_response_to", "status", "message"],
  "properties": {
    "action": {
      "type": "string",
      "description": "The action type of the response"
    },
        "in_response_to": {
      "type": "string",
      "format": "uuid",
```

```
      "description": "The message_id of the request being responded to"
    },
    "status": {
      "type": "string",
      "enum": ["success", "error"],
      "description": "Status of the response"
    },
    "message": {
      "type": "string",
      "description": "Human-readable description, especially for errors"
    }
  }
}
```