RFC for UPS/Amazon Communication Protocol

Group # 2
~~04/09/2019~~
04/24/2019
Committee Members:
Wenhui Guo, John Martins, Zizhao Fang, Naixin Yu,
Zhihao Gong, Rijish Ganguly, Yijie Yan, Xiaodi Nie

Abstract

This document offers a high-level introduction to the UPS Amazon Communication Protocol (UACP). Outlining the operations and requirements of UACP as a guide for users implementing UPS/Amazon communications streams.

Table of Contents

Introduction

The UPS Amazon Communication Protocol (UACP) is a new protocol, designed to enable consistent bidirectional communications between UPS and Amazon layers within an application world. UACP is built on top of the Google Protocol Buffer, a language and platform neutral mechanism for serializing structured data. This RFC outlines the Protocol Data Model, including both data format and operational workflow for UACP, as well as a set of Protocol Operations. Appendix A additionally provides an example Proto file for use with the Google Protocol Buffer.

This is a bidirectional communications model, however, not all operations are required to be understood from both sides for correct, standard operation, as, for example, only UPS can validly communicate a delivery as complete.

Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Terminology

- Google Protocol Buffer: A method used to generate structured data for communication protocols. A detailed introduction can be found in this document. In this project, Google Protocol Buffer is used to generate the messages sending between the UPS and Amazon servers.
- Sender: The server which sends the message. It MAY be the UPS server or the Amazon server.
- Receiver: The server which receives the message. It MAY be the UPS server or the Amazon server.
- Sequence Number: The unique identifier for each request and its corresponding response.

Protocol Model

1. Data Format

The message sent between the UPS server and the Amazon server MUST use the Google Protocol Buffer Message Format, which follows the rules described in this guide file. An example proto file is offered in the appendix.

2. Message Passing Workflow

There are four situations when the UPS server and the Amazon server need to communicate with each other:

- **Package Pickup:** When the Amazon server needs to ask the UPS to send a truck to pick up a package, it SHOULD send a PickUpPackage message to the UPS server. This message includes order id, warehouse address, ups account, product name and destination which are REQUIRED. And after the package is picked up, the UPS server SHOULD send a PickUpResponse message to the Amazon server to indicate the completion/failure of the pickup action. This message includes order id and track number which are REQUIRED.
- **Package Delivery:** When a package is packed, the Amzon server SHOULD send a Delivery Request message to the UPS server to pickup the package. When the package is delivered, the UPS server SHOULD send a ~~DeliveryComplete~~ Delivery Response message to the Amazon server to indicate the completion/failure of the delivery action. This message includes track number which is REQUIRED.
- **Address Updating:** When the user with an account updates the destination address of a product delivery task, the UPS server SHOULD send an AddressUpdate message to the Amazon server to indicate the changes. No response message except for the ACK message is needed for the AddressUpdate message.
- ~~**Status Query:** When the Amazon server wants to check the status of a package, it SHOULD send a StatusCheck message to the UPS server. And the UPS server SHOULD response with a StatusUpdate message to describe the current status of the corresponding package. Besides, the UPS server SHOULD also send a StatusUpdate message to the Amazon server every time there is a change on a package's status.~~

During each of these four processes, the UPS/Amazon server(sender) SHOULD send a request to the Amazon/UPS server(receiver) to indicate the action it wants to perform and offer the corresponding information, which will be covered in detail in the next section.

In order to avoid message lost, a unique sequence number, which is generated by the sender, MUST be attached with every request. And once the request is received, the receiver MUST send back an ACK with the sequence number of the received message to indicate that the message has been received. Multiple ACKs MAY be packed into one Google Protocol Buffer Message. If no ACK received, the sender SHOULD resend the request message until an ACK is received or timeout. And the sender SHOULD NOT assume that the receiver receives the message until getting the ACK message.

Protocol Operations

0. Connection setup

Amazon part:

1. set up a AMA_UPS (22222) server thread to listen to dedicated connection from UPS. Then receive U2AConnect from UPS, which contains worldid.
2. use the worldid from 1st step to send a AConnect message to WOLRD_AMA (23456). Then receive a AConnected from Amazon.
3. after 2nd step, send a U2AConnected message back to UPS, using AMA_UPS(22222).

Attention: step 1 2 3, order matters! step 4 5, order doesn't matter.

In the meantime:

4. set up a AMA_FRONT (33333) server thread to listen to a dedicated connection from Amazon front end. It depends on your design.
5. connect UPS_AMA (44444) and send request to UPS. Then ONLY send A2URequest and ONLY receive U2AResponse

After step2, WOLRD_AMA (23456) connection ONLY send ACommands and ONLY receive AResponses.

After step3, AMA_UPS(22222) connection ONLY receive U2ARequest and ONLY send back A2UResponse.

UPS part:

1. send UConnect message to world WORLD_UPS(12345) server without worldid. Then receive UConnected from world.
2. use the worldid from 1st step to send a U2AConnect to AMA_UPS(22222) server. Then receive a U2AConnected from Amazon

Attention: step 1 2, order matters! step 3 4, the order doesn't matter.

In the meantime:

3. set up a UPS_AMA (44444) server thread to listen to dedicated connection from Amazon. ONLY receive A2URequest and ONLY send back U2AResponse.
4. set up a UPS_FRONT (55555) server thread to listen to dedicated connection from UPS front end. It depends on your design.

After step1, WOLRD_UPS (12345) connection ONLY send UCommands and ONLY receive UResponses.
After step2, AMA_UPS(22222) connection ONLY send U2ARequest and ONLY receive A2UResponse.

In case of **any connection fail**, we MUST reset the server socket or reconnect the server from client side until it is successfully set up. **Attention**: if you find you recv a 0 length bytes from the socket, it indicates that this socket connection is broken. You MUST reconnect if it is a client or accept if it is a server. For more details, please refer to https://docs.python.org/3/howto/sockets.html

1. Pick Up Product

When the Amazon server needs to ask the UPS to send a truck to pick up a package, it SHOULD send a PickUpPackage message to the UPS server. The message MUST contain a sequence number orderid to identify a message ~~package,~~ an order id to trace back an Amazon order, a product id to indicate the product type, a product name, an address to indicate the destination of the package. Also, the message MAY optionally contain an ups account id to associate the package with a valid ups account, a priority number to denote the priority of the package.

2. Pick Up Complete

When the pick up action for one package is completed, UPS MUST inform Amazon the completion of picking up the package. The message MUST contain a sequence number which is used to identify a package, status which indicates success or failure, tracking number. And the message contains this package's next destination which is OPTIONAL.

3. Delivery Request

When the Amazon server received the Aloaded message from the world server, it SHOULD send a delivery request message to indicate the UPS side to go to delivery the package. The message MUST contains a seqnum to identify this message and a trucknum to indicate the order to be delivered.

4. ~~Delivery Complete~~ Delivery Response

When the delivery of one package is completed regardless of success or failure, UPS MUST inform Amazon the complete of each package. The message MUST contain a sequence number, its tracking number. After Amazon receives this message, it MUST send an ACK back to UPS with a sequence number.

5. ~~Update Delivery Details~~ UpdateDest

When the destination address of one product delivery task needs to be changed, UPS MUST inform Amazon about the update. The ~~"AddressUpdate~~UpdateDest" message as mentioned above MUST contain a sequence number, the order number of this product delivery task, and the new address. The ACK for this message MUST be sent from Amazon to UPS containing the aforementioned sequence number to indicate that the message is received.

6. Error Message

In case any error happens during this process, both the UPS server and the Amazon server COULD send Error Messages to the other side. It MUST contain a originseqnum to indicate which message cause the error and a seqnum to identify this message and a string to describe the actual error.

~~5. Status Update~~

~~Whenever Amazon makes a request to UPS with a particular track number and sequence number which were generated earlier, UPS MUST respond with the status of the package. The package can be on route to the destination, getting picked up from the warehouse or in the process of being picked up. If an error occurs on the delivery or something goes wrong, an appropriate error message should be sent back to UPS in response to the status query. In addition to that, every time the status of package changes, UPS MAY inform Amazon about the change using the appropriate track and sequence num.~~

~~6. Query Status~~

~~When Amazon sends a request to the UPS to retrieve order status, the tracknum, sequence num of the delivery MUST be included in the request. Then, UPS MUST send an acknowledgment to Amazon to confirm the request. After processing the request. UPS will send the response, which would be in the same format of the "StatusUpdate" message, back to Amazon. After Amazon receives the response, an acknowledgment will be sent back to UPS. Afterward, Amazon will display the status to the customer of the package.~~

Appendix A: Example Proto File

```
syntax = “proto2”;

message U2AConnect{
   required int64 worldid = 1;
}

message U2AConnected{
   required int64 worldid = 1;
   required string result = 2;
}

message PickupRequest{
      required int64 seqnum = 1;
      required int64 orderid = 2;
      required string productName = 3;
      required string warehouseAddress = 4;
      required string destination = 5 4;
      required int64 whid = 4;
      required int64 wh_x = 5;
      required int64 wh_y = 6;
      required int64 dest_x = 7;
      required int64 dest_y = 8;
      optional string upsAccount = 6 9;
}

message PickupResponse{
      required int64 seqnum = 1;
      required int64 tracknum = 2;
      required int64 orderid = 3;
      required int64 truckid = 4;
}
message DeliveryRequest {
      required int64 seqnum = 1;
      required int64 tracknum = 2;
}
message DeliveryResponse{
      required int64 seqnum = 1;
      required int64 tracknum = 2;
}
```

```
message UpdateDest{
	required int64 seqnum = 1;
	required int64 tracknum = 2;
	required int64 new_x = 3;
	required int64 new_y = 4;
	required string newDest = 3;
}

message ErrorMessage{
	required string err = 1;
	required int64 originseqnum = 2;
	required int64 seqnum = 3;
}

message A2URequest {
	repeated  PickupRequest pickup = 1;
	repeated  DeliveryRequest delivery = 2;
	repeated  int64 ack = 3;
}

message U2AResponse {
	repeated PickupResponse pickup = 1;
	repeated DeliveryResponse delivery = 2;
	repeated ErrorMessage error = 3;
	repeated int64 ack = 4;
}

message U2ARequest {
	repeated UpdateDest dest = 1;
	repeated int64 ack = 2;
}

message A2UResponse {
	repeated ErrorMessage error = 1;
	repeated int64 ack = 2;
}
```