hopper

Hotel Message Protocols

Version 1.1

November 29, 2011

A. Transport

Hopper's core transport protocol is **AMQP** messaging. We are using the open-source RabbitMQ implementation of AMQP internally. A commercially-supported version of RabbitMQ is available from VMWare SpringSource.

For message serialization, we use **ProtocolBuffers**. You can find out more here: http://code.google.com/apis/protocolbuffers/. It is simple, language-agnostic and much more IO-efficient than XML, which is important considering the message volume that we are dealing with.

B. Messages

For hotel data, Hopper currently supports two types of message: **HotelStay** and **HotelDescription**, described in detail over the following pages.

C. Consistency

Dropped messages are considered acceptable. They result in corresponding offers being temporarily stale or unavailable in the Hopper search results. In the case of excessive message backlog due to communication failure, it is recommended that messages be dropped. It is recommended that any message buffer have a maximum size of no more than 1 million messages before truncation.

D. Triggers

The Hopper hotel protocol is a **Push API**, which implies that the data provider must initiate messages according to its own triggering logic.

HotelDescription messages can simply be triggered once per week per hotel. Note that a hotel cannot appear in Hopper search results until a HotelDescription message has been sent at least once.

The recommended strategy for triggering **HotelStay** messages is as follows:

```
For each hotel
    For each check-in date between today and today+180 days
        For each length of stay from 1 day to 14 days
            Trigger one message for each rate+room offer:
            - every 3 hours if check-in date is < 30 days from today
            - every 6 hours if check-in date is >30 days and < 60 days from today
            - every 12 hours if check-in date is > 60 days from today
```

The effective message rate will be about **0.39 msgs / sec / hotel**, assuming 4 rate+room offers per hotel:

```
4 HotelStays x ( (8 updates /day x 30 check-in dates x 14 lengths of stay) +
                 (4 updates/day x 30 check-in dates x 14 lengths of stay) +
                 (2 updates/day x 120 check-in dates x 14 lengths of stay) )
            4 x (3,360 / day + 1,680 / day + 3,360 / day)
            = 33,600 / day = 1,400 / hour = 23.33 / minute = 0.39 / sec
```

HotelStay

The HotelStay message represents a bookable continuous sequence of nights for a single room at a single lodging property.

Fieldname	Type	Req	Collection	Description
property_id	string	yes		The unique identifier of the hotel in the merchant system
checkin_date	long	yes		The check-in (arrival) date/time of this stay. A 64-bit long representing milliseconds since midnight, January 1, 1970 UTC.
checkout_date	long	yes		The check-out (departure) date/time of this stay. A 64-bit long representing milliseconds since midnight, January 1, 1970 UTC.
room_type	string			The name of room type for this stay (e.g. “Deluxe King” or “Standard Double”)
rate_type	string			The name of the rate type for this stay (e.g. “Internet Rate”, or “Honeymoon Special”)
base_amount	double	yes		The average nightly base amount for this hotel stay. If the rate varies over the stay period, then this amount represents the total cost for the entire stay period divided by the number of nights.
tax_amount	double	yes		The average nightly tax amount (all taxes and fees) for this hotel stay
currency_code	string	yes		The international currency code for the base_amount and tax_amount values.
merchant_id	string	yes		The unique ID of the merchant where this stay is offered for sale (4 alpha characters assigned by Hopper)
timestamp	long	yes		The time that this stay was offered for sale. A 64-bit long representing milliseconds since midnight, January 1, 1970 UTC.
booking_path	string	yes		The URL where this stay can be booked. Hopper will refer consumers to this path to purchase the stay.

HotelDescription

The HotelDescription message represents the name, address and other static attributes of a lodging property for which a merchant holds inventory. The purpose of this message is to match a merchant hotel id (property_id) to Hopper's own database of hotels. The description fields provided, then, serve as clues to unambiguously identify the hotel. This information will not be displayed to users, unless it matches information already in the Hopper system.

Fieldname	Type	Req	Collection	Description
merchant_id	string	yes		The unique ID of the merchant where this hotel is offered for bookings (4 alpha characters assigned by Hopper)
property_id	string	yes		The unique identifier of the property in the merchant system (e.g. 1342143)
name	string	yes		The name for this lodging property in the merchant system (e.g. “The Drake Hotel”)
brand	string			The brand name of this property, if available (e.g. Sheraton)
star_rating	integer			The star rating of this hotel (number of stars)
street_address	string	yes		The official street address of this lodging property (e.g. 123 Main St.)
city	string	yes		The city where this property is located (e.g. Boston)
state_province	string			The full name or code of the state or province where this property is located (e.g. Massachussetts or MA)
country_code	string	yes		The ISO 3166-1 country code of the country where this propety is located (See http://en.wikipedia.org/wiki/ISO_3166-1)
postal_code	string			The zip or postal code of the property
telephone	string	yes		The telephone number (including country and area code) of the property (not the chain), without separators.
longitude	double			The longitude of the property (WGS84)
latitude	double			The latitude of the property (WGS84)

HotelProtocols: hotels.proto

```
package hotels;

option java_package = "com.hopper.models.hotels";
option java_outer_classname = "HotelProtocols";

message HotelStay {
    required string property_id = 1;
    required sfixed64 checkin_date = 2;
    required sfixed64 checkout_date = 3;
    optional string room_type = 4;
    optional string rate_type = 5;
    required double base_amount = 6;
    required double tax_amount = 7;
    required string currency_code = 8;
    required string merchant_id  = 9;
    required sfixed64 timestamp  = 10;
    required string booking_path = 11;
}

message HotelDescription {
    required string merchant_id = 1;
    required string property_id = 2;
    required string name = 3;
    optional string brand = 4;
    optional int32 star_rating = 5;
    required string street_address = 6;
    required string city = 7;
    optional string state_province = 8;
    required string country_code = 9;
    optional string postal_code = 10;
    required string telephone = 11;
    optional double longitude = 12;
    optional double latitude = 13;
}
```