DFHV integration interface specification

TECHNICAL SPECIFICATION V2, Document Version 1.15

AUTHORED BY: SOLVD INC

Table of Contents

Project overview

DFHV serves as an obligatory “umbrella” organization for all taxi services in DC. An integrated data hub will collect data from all providers including but not limited to DTS and PSP to enable better vehicle tracking using mobile apps, digital meters. The system will track location of the vehicles available for hire and trips.

Purpose of the document

The document describes, architecture and service interface that will allow taxi companies to exchange information with DFHV Integration Data Hub (the system).

The information contained in this document is subject to change without notice in order to improve reliability, design or function and does not represent a commitment on the part of DFHV.

Audience

This document is intended for programmers building applications that interact with DFHV Integration Data Hub. This document assumes that the audience is aware of the terminology associated with reporting vehicle, trip and driver information to the DFHV.

Contact Information

If you have any question related to the API usage or format of the data, please contact DFHV Data Hub Support at dfhv.datahubsupport@dc.gov.

Architecture overview

The system is built on TRANSITiQ platform by extending existing capabilities and integrating with DC Taxi app. TRANSITiQ is built on top of the Azure Cloud infrastructure and provides reliable data collection and processing capabilities.

The system will consist from the following components:

- Azure Event Hubs is a highly scalable publish-subscribe service that will allow to ingest millions of vehicle location events per second and stream them into multiple internal data processing components.
- Real-time analytics providers and batch processing services will transform, process and store the data for immediate and future consumption.
- Vehicle and trip updates data will be communicated using rest API services then normalized and stored in the Azure SQL database.
- Azure Enterprise Service bus to will be used for communication between taxi companies and the system. The service bus will expose a dedicated line of communication with the system (topic). Meter providers and taxi companies will subscribe for dedicated topics and be notified of newly created eHail requests or other updates in the system.
- Azure storage to keep vehicle, trips, location and other historical information.
- Azure App Service will serve data API for DFHV Taxi mobile application. The system will use vehicle location information to suggest closest taxi cab to users and generate hails. Hails will be routed to Taxi companies according to business rules.

Sandbox environment

To sign up for the sandbox environment developers need to register using the API portal at https://dfhv-api-documentation.terraiq.io-api.net/ and request access to the DFHV-api-dev.

The portal contains detailed schema and type specification for the API input and output parameters. As part of the registration process, developers will receive ClientId and ClientKey parameters. These parameters should be passed to Access endpoints to request appropriate connection information for Event Hub and ESB Subscription.

Available API endpoints to request connection information:

- Returns access details for the event hub to send location updates

Location and status updates

Azure Event Hub

Location and status updates data should be sent to the dedicated Azure Event Hub. Every Meter provider will need to request a unique URI that will be secured by a per-publisher token, so that only the Meter provider company that poses the right token can publish event to this endpoint. To get Event Hub connection information, send GET request to GetGeolocationUpdateEndpoint. Provide mandatory ClientId and ClientKey parameters to receive connection information.

The developer guide provides detailed information on sending the data to Event Hubs.

Event data format

Location and status update events to be reported in real-time using the following JSON format:

```
[
      {
          “t”: “2016-06-07T10:38:27.340Z”, // event timestamp
          “pvin”: “A123”, // PVIN of the Taxi
          “did”: “ABC123”, // Driver ID = FaceCardID
          “lt”: 0, // Latitude
          “ln”: 0, // Longitude
          “m”: 1, // Meter status [0 = On, 1 = Off]
          “s”: 0  // Availability status [1 = Available, 2 = OnCall, 3 = Hired, 4 =
OffDuty, 5 = Emergency]
      }
]
```

Trip updates, Availability Statuses and Assignments (DFHV Integration Data Hub API)

Each Provider (PSP or DTS) will be assigned a ProviderCode. And each Certified Taxi Meter will be assigned a TaxiMeterCode.

So, there is a combination of ClientID (APIClientID), ProviderCode and TaxiMeterCode must be provided with each trip submission. ClientID identities actual client of API portal, Provider code – the provider that completed the trip and Taxi Meter – identifies the taxi meter the trip has been made with.

Each API Method on API Portal contains comprehensive information about possible outcomes of calling the method as well as well-defined schema of each request.

Driver updates

Available API endpoints for the Driver record updates:

- Assign driver to vehicle

Additional information for submitting Assignments:

Field Name	Description	Type	Required

PVIN	The PVIN of the vehicle	String 20	Yes, always
DecalNumber	The Decal Number for None District Limos (NDL) vehicles	String 20	No, submit NULL
FacecardID	The Face Card ID of District taxi or District Limo driver	String 20	Yes, always
DriverID	The Driver ID of the NDL drivers	String 20	No, submit NULL
Timestamp	Time and Date when assignment happened	ISO 8601 formatted datetime in UTC, example: 2017-11-20T19:32:23Z	No, will be set to the current time if not provided

Vehicle updates

Available API endpoints for the Vehicle record updates:

- Update availability status for the vehicle

Additional information for submitting Availability Statuses:

Field Name	Description	Type	Required
PVIN	The PVIN of the vehicle	String 20	Yes, always
DecalNumber	The Decal Number for None District Limos (NDL) vehicles	String 20	No, submit NULL
FacecardID	The Face Card ID of District taxi or District Limo driver	String 20	Yes, always
DriverID	The Driver ID of the NDL drivers	String 20	No, submit NULL
Status	The availability status value	• Available = 1 • OnCall = 2 • Hired = 3 • OffDuty = 4 • Emergency = 5	Yes
Timestamp	Time and Date when assignment happened	ISO 8601 formatted datetime in UTC, example: 2017-11-20T19:32:23Z	No, will be set to the current time if not provided

Trip updates

Available API endpoints for the Trip updates:

- Register new trip
- Update previously created trip
- Update previously created trip by external ID
- Upload receipt
- Upload receipt by externalID
- Upload Receipt without meter code
- Upload Receipt by external ID without meter code Additional information for submitting trips:

Field Name	Description	Type	Required
`ExternalID`	The unique ID from their party system. The ID needs to be unique within the scope of TaxiMeterCode and ProviderCode	String 50	Yes, for all trip statuses
`ExternalID2`	For future use	String 50	No
`PVIN`	The PVIN of the vehicle	String 20	Yes, always
`DecalNumber`	The Decal Number for None District Limos (NDL) vehicles	String 20	No, submit NULL
`FacecardID`	The Face Card ID of District taxi or District Limo driver	String 20	Yes, always
`DriverID`	The Driver ID of the NDL drivers	String 20	No, submit NULL
`Type`	Defines the trip type.	• Ordinal = 1 • General = 1 • VoD = 2 • TransportDC = 3 • TransportDCShared = 4 • MOVA = 5 • CFSA = 6, • NRS = 7 • NEMT = 8 • T2R = 9 • SME = 10 • DBH = 11	Yes, when the trip status is set to “Completed” or “Canceled”
`Rating`	The trip rating	Integer	No, submit if applicable

PassengerNum	The number of passengers	Integer	Yes, when the trip status is set to “Completed”
StopNum	The number of stops for shared ride	Integer	Yes, when the trip status is set to “Completed”. Set it to ONE (1) for none shared rides.
StartDateTimeUTC	The timestamp when the trip started in UTC	Datetime UTC ISO 8601 formatted datetime in UTC, example: 2017-1120T19:32:23Z	Yes, always
EndDateTimeUTC	The timestamp when the trip ended in UTC	Datetime UTC ISO 8601 formatted datetime in UTC, example: 2017-1120T19:32:23Z	Yes, when the trip status is set to “Completed”

Duration	Duration of trip in seconds	Integer	Yes, when the trip status is set to “Completed”
EhailID	eHail ID associated with trip	GUID	No
Milage	The total distance in miles	Double	Yes, when the trip status is set to “Completed”
Origin	The pickup location. Address and Latitude and Longitude	String • Street Number 20 • Street name 150 • City 100 • State 2 • Zip 5 Double • Latitude • Longitude	Yes, when the trip status is set to “Completed” or “Canceled”

Destination	The drop off location. Address and Latitude and Longitude	String • Street Number 20 • Street name 150 • City 100 • State 2 • Zip 5 Double • Latitude • Longitude	Yes, when the trip status is set to “Completed”
FareAmount	The fare amount	Double	No, submit if applicable
ExtraFareAmount	The extra fare fee	Double	No, submit if applicable
DispatchAmount	The dispatch fee	Double	No, submit if applicable
EmergencyAmount	The emergency fee	Double	No, submit if applicable
AirportAmount	The airport fee	Double	No, submit if applicable
PassengerAmount	The extra passenger fee	Double	No, submit if applicable
GratuityAmount	The gratuity amount	Double	No, submit if applicable
SurchargeAmount	The DFHV surcharge amount	Double	No, submit if applicable
TollAmount	The toll fee	Double	No, submit if applicable
TotalAmount	The toll amount of all other fields	Double	No, submit if applicable
DiscountAmount	The discount amount	Double	No, submit if applicable
FareType	The type of fare.	• General = 1 • Shared = 2 • Hourly = 3 • Flat = 4 • Upfront = 5	Yes, when the trip status is set to “Completed”.

PaymentType	The payment type.	• Credit = 1 • Cash = 2 • EHail = 3	Yes, when the trip status is set to “Completed”
Status	The status of trip.	• InProgress = 1 • Pending = 2 (Set automatically when Client tried to submit incomplete information for “Completed” trip.) • Completed = 3 • Canceled = 4	Yes
RiderCode	The unique identifier code for rider. This can be used for DFHV special programs.	String 50	No
RideRequestCode	Code used to request a ride under Transport DC program terms	String (4), alphanumerical	No
IsWavRequested	Flag that indicates Wheelchair Accessible Vehicle (WAV) was requested	Boolean	No
CallDateTimeUTC	The date and time when scheduled pickup was requested	Datetime UTC ISO 8601 formatted datetime in UTC, example: 2020-1020T19:27:23Z	No
ScheduledPickUpDateTimeUTC	The date and time pickup are scheduled	Datetime UTC ISO 8601 formatted datetime in UTC, example: 2020-1023T19:32:23Z	No
HailType	Describes how ride request was originated	• StreetHail = 0 • EHail = 1 • PhoneCall = 2	No

IsCapExemption	Flag that indicates that given trip is exempt from regular Cap	Boolean Accepted values: true, false	No

- You may submit the trip data once all together (Trip_Create) or in multiple attempts (first Trip_Create, then Trip_Update)
- If you submit trip data with status “Completed” and some of the requierd values are missing system will save/update the submitted trip data with status “Pending”.
- Your have SEVEN (7) days to update the trip data after it has been created.
- For DecalNumber and DriverID submit NULL.
- ExternalID must be unique within the scope of ProviderCode and TaxiMeterCode, Data Integration Hub will not accept two trips with the same ExternalID
- FacecardID & PVIN are mandatory.
- For Origin and Destination: Latitude, Longitude, City, State, StreetName, StreetNumber and Zip must be set.

Update frequency requirements

- Taxi Meter providers are required to submit vehicle location and status updates for every active vehicle in the fleet with 3 seconds interval. If technical limitations are in place to comply with this requirement, the provider representative needs to notify DFHV and provide technical specifications for the best possible update frequency.
- Taxi Meter providers should not send vehicle location and status updates when the driver assigned to the vehicle is off duty.
- Driver assignments and availability changes need to be submitted as they happen within 20 seconds from the time it happened.
- Trip information updates need to be submitted when trip starts and when trip completes, within 20 seconds from the time it happened.

Verification API

DFHV provide endpoint to verify whether driver and vehicle are able to provide their services in DC. The API called Verification API exposes 4 methods:

1. Verify a pair of PVIN and Face ID
2. Get Driver Info
3. Get Vehicle Info

The verify method must be used every time driver/vehicle start their shift/login into their Digital Taxi Meter.

Document Version History

Version	Date	Author	Description
V1.0	05/10/2016	Vladimir Bychkov	Initial draft release
V1.1	06/07/2016	Vladimir Bychkov	Published API V1
V1.2	06/09/2016	Vladimir Bychkov	Minor updates
V1.3	07/11/2016	Vladimir Bychkov	Added “Update frequency requirements”
V1.4	11/17/2017	Dmitriy Reshetov	Reworked to meet API v2 requirements
V1.5	12/25/2019	Vladimir Bliznikov	Updated with new TRIP fields
V1.6	04/02/2020	Vladimir Bliznikov	Add new trip type value – T2R
V1.7	09/15/2020	Vladimir Bliznikov	Add new trip type value – SME
V1.8	10/6/2020	Vladimir Bliznikov	Updated with new TRIP fields: CallDateTimeUTC, ScheduledPickUpDateTimeUTC
V1.9	06/30/2021	Vladimir Bliznikov	Updated with new Trip field: HailType
V1.10	07/29/2021	Vladimir Bliznikov	Update with new Trip field: IsCapExemption
V1.11	09/14/2022	Vladimir Bliznikov	NDL Verification endpoint is added to the document
V1.12	11/06/2024	Vladimir Bliznikov	Add new trip type value – DBH
V1.13	11/20/2024	Hanna Slapik	Created new Fare Type: Upfront. Crated Trip Type: General as a synonym to Ordinal. Removed Payment Types: Uber and Other
V1.14	03/04/2025	Hanna Slapik	Updated ExtraFareAmount description in API documentation
V1.15	03/27/2025	Hanna Slapik	Update api links, remove deprecated Verification API endpoints