TripThru

GATEWAY API v1.0

API ENDPOINTS

ACCESS

Once registered you will receive an access token that follows OAuth2.0. This will grant controlled access for partnered dispatch systems on behalf of TripThru. See Authentication for details.

SET NETWORK INFO

Set network's information. This is how TripThru knows about a new network's information. Since all networks support the "Get network info" API call, it is through this method that the network details are revealed. If the chosen interface is RESTful the network needs to provide a callback URL and TripThru will authenticate to this endpoint using the same access token the network uses to authenticate to TripThru.

RESTful	socket.io*
POST /network	set-network-info

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "name": "Fabless",
    "callback_url": "https://api.grabtaxi.com/tripthru",  //RESTful only
    "products": [
        {
            "id": "Fabless Los Angeles",
            "name": "Fabless Los Angeles",
            "image_url": "http://www.tripthru.com/assets/networks/fabless.png",
            "accepts_prescheduled": true,
            "accepts_ondemand": true,
            "accepts_cash_payment": true,
            "accepts_account_payment": true,
            "accepts_creditcard_payment": true,
            "coverage": {
                "radius": 50,
                "center": {
                    "lat": 34.0494,
                    "lng": -118.242
                }
            }
        },
        {
            "id": "Fabless New York",
            "name": "Fabless New York",
            "image_url": "http://www.tripthru.com/assets/networks/fabless.png",
            "accepts_prescheduled": true,
            "accepts_ondemand": true,
            "accepts_cash_payment": true,
            "accepts_account_payment": true,
            "accepts_creditcard_payment": true,
            "coverage": {
                "radius": 50,
                "center": {
                    "lat": 40.6711,
                    "lng": -73.9404
                }
            }
        }
    ]
}
```

Name	Type	Description
id	string	Network id
name	string	Network name
callback_url	string	**WebHook callback URL (for RESTful request only)**
products[n]	object	Product objects
products[n].id	string	Product unique identifier
products[n].name	string	Product display name
products[n].image_url	string	Product image URL
products[n].capacity	integer	Max passenger count that this product can accommodate
products[n].accepts_prescheduled	Boolean	Whether product accepts prescheduled trips
products[n].accepts_ondemand	Boolean	Whether product accepts on-demand trips
products[n].accepts_cash_payment	Boolean	Whether product accepts cash payment
products[n].accepts_account_payment	Boolean	Whether product accepts account payment
products[n].accepts_creditcard_payment	Boolean	Whether product accepts creditcard payment
products[n].coverage	list	Zone objects list
products[n].coverage[n].center	object	Zone location object
products[n].coverage[n].center.lat	double	Coverage latitude
products[n].coverage[n].center.lng	double	Coverage longitude
products[n].coverage[n].radius	double	Coverage radius in km

RESPONSE

If successful, this method returns a response body with the following structure:

```
{
    "result": "OK",
    "result_code": 200
}
```

Name	Value	Description
result	"OK"	
result_code	200	

GET NETWORK INFO

Get a network's information. This method is used by TripThru to update the required network information.

RESTful	socket.io*
GET /network/:id	get-network-info

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "fabless@tripthru.com",     // socket.io* only
}
```

RESPONSE

If successful, this method returns a response body with the following structure:

```
{
    "name": "GrabTaxi",
    "product": [
        {
            "id": "11lk34",
            "name": "Taxi",
            "zone": [
                {
                    "location": {
                        "lat": 52.12588,
                        "lng": 11.61150
                    },
                    "radius": 200.0
                }
            ],

        },
        {
            "id": "11lk34",
            "name": "Blackcar",
            "zone": [
                {
                    "location": {
                        "lat": 22.12588,
                        "lng": 11.61150
                    },
                    "radius": 100.0
                }
            ],

        }
    ],
    "result_code": 200,
    "result": "OK"
}
```

Name	Type	Description
result	"OK"	
result_code	200	
name	string	Network name
products	list	Product objects list

products[n]	object	Product object
products[n].id	string	Product unique identifier
products[n].name	string	Product display name
products[n].image_url	string	Product image URL
products[n].accepts_prescheduled	Boolean	Whether product accepts prescheduled trips
products[n].accepts_ondemand	Boolean	Whether product accepts on-demand trips
products[n].accepts_cash_payment	Boolean	Whether product accepts cash payment
products[n].accepts_account_payment	Boolean	Whether product accepts account payment
products[n].accepts_creditcard_payment	Boolean	Whether product accepts creditcard payment
products[n].coverage	list	Zone objects list
products[n].coverage[n].center	object	Zone location object
products[n].coverage[n].center.lat	double	Coverage latitude
products[n].coverage[n].center.lng	double	Coverage longitude
products[n].coverage[n].radius	double	Coverage radius in km
products[n].cancellation_fee	double	Product cancellation fee

GET QUOTE

Get a quote's current state. Requests a list of quotes (fare, ETA, products, etc.).

This method may be called before dispatching. It determines availability, fare, and many other things.

RESTful	socket.io*
GET /quote	get-quote

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "customer": {
       "id": "51fbc4fea"
    },
    "pickup_location": {
        "lat": 52.12588,
        "lng": 11.61150
    },
    "dropoff_location": {
        "lat": 52.5373399193,
        "lng": 13.378729824
    },
    "pickup_time": "2013-08-21T14:42:46Z",
    "payment_method_code": "account",
    "product_id": "taxi",
    "passengers": "2",
    "luggage": "3"
}
```

Name	Type	Description

customer_id	string	Customer ID for (Customer experience optimization) (optional)
pickup_time	timestamp	GMT time of the pickup.
pickup_location	object	Pickup location object
pickup_location.lat	double	Pickup location latitude
pickup_location.lng	double	Pickup location longitude
dropoff_location	object	Dropoff location object (optional)
dropoff_location.lat	double	Dropoff location latitude (optional)
dropoff_location.lng	double	Dropoff location longitude (optional)
payment_method_code	string	Can be "account", "cash" or "credit-card"
product_id	string	Product unique identifier filter (optional)
passengers	integer	Number of passengers
luggage	integer	Amount of luggage
cancellation_fee	double	Product cancellation fee

RESPONSE

If successful, this method returns a response body with the following structure. Note that all fields may not be returned.

```
{
   "quotes": [
       {
            "id": "1234",
            "network": {
               "id": "gettaxi@tripthru.com",
               "name": "GetTaxi"
            },
            "product": {
                "id": "1234",
                "name": "Taxi"
            },
            "fare": {
                "estimate": "12.5",
                "low_estimate": "11.5",
                "high_estimate": "13.0",
                "currency_code": "USD"
            },
            "eta": "2015-01-26T14:36:03.337Z",
            "duration": "875",
            "distance": "5.5"
        }
   ],
   "result": "OK",
   "result_code": 200
}
```

Name	Type	Description
result	"OK"	
result_code	200	
quotes	list	Quote objects list
quotes[n]	object	Quote objects
quotes[n].id	string	Quote unique identifier
quotes[n].network	object	Network
quotes[n].network.id	string	Network unique identifier
quotes[n].network.name	string	Network name
quotes[n].product	object	(optional)

quotes[n].product.id	string	Product unique identifier
quotes[n].product.name	string	Product name
quotes[n].product.image_url	string	Product image URL
quotes[n].product.cancellation_fee	double	Product cancellation fee
quotes[n].price	object	Estimate of trip fare
quotes[n].price.estimate	string	Estimate of trip fare.
quotes[n].price.low_estimate	float	Lower bound on trip fare
quotes[n].price.high_estimate	float	Upper bound on trip fare
quotes[n].price.currency_code	float	Currency code
quotes[n].eta	timestamp	Estimated pickup time (UTC)
quotes[n].distance	float	Estimated trip distance (km)
quotes[n].duration	float	Estimated trip duration (seconds)

DISPATCH TRIP

Request dispatch. Dispatches a trip to a fleet. This method can be used in conjunction with "Get Trip Quotes" but is not required.

RESTful	socket.io*
POST /trip/:id	dispatch-trip

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "12@tripthru.com",      // socket.io* only
    "customer": {
       "name": "Joe",
       "local_id": "en_US",
       "phone_number": "233-553-2343"
    },
    "pickup_location": {
        "lat": 25.037749,
        "lng": 121.536265,
        "description": "will be waiting at main entrance."
    },
    "pickup_time": "2015-01-26T14:22:50.018Z",
    "dropoff_location": {
        "lat": 25.037234,
        "lng": 121.532006
    },
    "product_id": "taxi",
    "passengers": "2",
    "luggage": "3",
    "payment_method_code": "account"
}
```

Name	Type	Description
customer	string	Customer object

customer.id	string	Unique identifier (to be used of experience optimization (optional)
customer.name	string	First name only (recommended)
customer.local_id	string	Language / local of customer (Java local ID)
customer.phone_number	string	Virtual (recommended)
passengers	integer	Number of passengers
luggage	integer	Amount of luggage
quote_id	string	Unique identifier of quote (optional)
network_id	string	Unique identifier of servicing network (optional)
product_id	string	Unique identifier of product (optional)
pickup_time	timestamp	Pickup time (UTC)
pickup_location	object	Pickup location object
pickup_location.lat	double	Pickup location latitude
pickup_location.lng	double	Pickup location longitude
pickup_location.description	string	Description of pickup location. This could just be the textual address.
dropoff_location	object	Dropoff location object (optional)
dropoff_location.lat	double	Dropoff location latitude
dropoff_location.lng	double	Dropoff location longitude
dropoff_location.description	string	Description of pickup location. This could just be the textual address (optional)
payment_method_code	string	Can be "account", "cash", or "credit-card"
tip	object	Guaranteed Tip (optional)
tip.amount	float	Tip amount
tip.currency_code	string	Currency code

RESPONSE

If successful, this method returns a response body with the following structure. Note that all fields may not be returned.

```
{
    "result": "OK",
    "result_code": 200
}
```

GET TRIP STATUS

Get trip's current status. Returns the current trip status and driver status (including location tracking) of a given trip. Note that not all networks will fully support tracking so the response fields are all optional. They will be supplied by the network / fleet to the degree that the network / fleet supports tracking.

RESTful	socket.io*
GET /tripstatus/:id	get-trip-status

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "123@fablesslabs.com"      // socket.io* only
}
```

RESPONSE

If successful, this method returns a response body with the following structure.

```
{
    "result": "OK",
    "result_code": 200,
    "status": "en_route",
    "eta": "2015-01-26T14:07:41.363Z",
    "fare": {
        "amount": 12.05,
        "currency_code": "USD"
    },
    "driver": {
        "id": "45399",
        "name": "Johnny",
        "local_id": "en_US",
        "location": {
            "lat": 37.979,
            "lng": 23.73153
        }
    }
}
```

Name	Type	Description
result	"OK"	
result_code	200	
status	string	• new • accepted • en-route • picked_up • dropped_off • no_show • completed • cancelled • rejected
eta	timestamp	ETA of driver to pickup
product	object	(optional)
product.id	string	Product unique identifier
product.name	string	Product display name
product.image_url	string	Product image URL (optional)
fare.amount	float	Payment amount to be confirmed by customer
fare.currency_code	string	Currency code
driver	object	Driver object (if available)
driver.id	object	Driver unique identifier (for driver and customer experience optimization) (optional)
driver.name	string	First name (recommended)
driver.local_id	string	Locale of driver
driver.native_language_id	string	Native language of driver (optional)
driver.location	object	Driver current location
driver.location.lat	object	Driver current location's latitude
driver.location.lng	object	Driver current location's longitude

driver.location.description	object	Textualization or description of current location (optional)

UPDATE TRIP STATUS

Updates the trip status information. Updates the trip status information, including driver's tracking information.

RESTful	socket.io*
PUT /tripstatus/:id	update-trip-status

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "12@tripthru.com",  // socket.io* only
    "status": "enroute",
    "eta": "2015-01-26T14:07:41.363Z",
    "fare": "12.05",
    "currency_code": "USD",
    "driver": {
        "id": "45399",
        "name": "Johnny",
        "local_id": "en_US",
        "location": {
            "lat": 37.979,
            "lng": 23.73153
        }
    }
}
```

Name	Type	Description
status	string	• new • accepted • en-route • picked_up • dropped_off • no_show • completed • cancelled • rejected
eta	timestamp	ETA of driver to pickup
product	object	(optional)
product.id	string	Product unique identifier
product.name	string	Product display name
product.image_url	string	Product image URL (optional)
fare	float	Payment amount to be confirmed by customer
currency_code	string	Currency code
driver	object	Driver object (if available)

driver.id	object	Driver unique identifier (for driver and customer experience optimization) (optional)
driver.name	string	First name (recommended)
driver.local_id	string	Locale of driver
driver.native_language_id	string	Native language of driver (optional)
driver.location	object	Driver current location
driver.location.lat	object	Driver current location's latitude
driver.location.lng	object	Driver current location's longitude
driver.location.description	object	Textualization or description of current location (optional)

RESPONSE

If successful, this method returns a response body with the following structure.

```
{
    "result": "OK",
    "result_code": 200
}
```

Name	Type	Description
Result	"OK"	
result_code	200	

GET DRIVERS NEARBY

Returns list of nearby drivers contained within partnered networks.

RESTful	socket.io*
GET /drivers	get-drivers-nearby

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "limit": 10,
    "radius": 0.1,
    "location": {
        "lat": 37.979,
        "lng": 23.73153
    }
}
```

Name	Type	Description
limit	int	Limit number of drivers (optional)
location	object	Search location
location.lat	float	Search location's latitude
location.lng	float	Search location's longitude

radius	float	Search radius in km. Default value is 0.1 (optional)
product_id	string	This filters drivers on product unique identifier (optional)

RESPONSE

If successful, this method returns a response body with the following structure.

```
{
    "count": 4,
    "drivers": [
        {
            "lat": "52.12724",
            "lng": "11.60905",
            "eta": "2015-01-26T14:36:03.337Z",
            "product": {
                "id": "1234",
                "name": "Taxi"
            }
        },
        {
            "lat": "52.12588",
            "lng": "11.61150",
            "eta": "2015-01-26T14:36:03.337Z",
            "product": {
                "id": "1234",
                "name": "Blackcar"
            }
        },
        {
            "lat": "52.12145",
            "lng": "11.61194",
            "eta": "2015-01-26T14:36:03.337Z",
            "product": {
                "id": "1234",
                "name": "Taxi"
            }
        },
        {
            "lat": "52.12961",
            "lng": "11.60787",
            "eta": "2015-01-26T14:36:03.337Z",
            "product": {
                "id": "1234",
                "name": "Taxi"
            }
        }
    ],
    "result": "OK",
    "result_code": 200
}
```

Name	Type	Description
result	"OK"	
result_code	200	
count	int	Number of drivers returned
drivers	list	Driver objects list
drivers[n].lat	float	Latitude of driver's current location
drivers[n].lng	float	Longitude of driver's current location
drivers[n].eta	timestamp	Estimated pickup time (UTC)

drivers[n].product	object	Product that this driver provides
drivers[n].product.id	string	Product unique identifier
drivers[n].product.name	string	Product display name
drivers[n].product.image_url	string	Product image URL

REQUEST PAYMENT

Confirm payment. After a trip reaches a status of complete, the servicing partner must post a payment transaction to be forwarded to the customer for confirmation.

RESTful	socket.io*
POST /payment/:id	request-payment

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "12@tripthru.com",   // socket.io* only
    "fare": "12.05",
    "currency_code": "USD",
    "fare_summary": "12.05"
}
```

Name	Type	Description
fare	float	Payment amount to be confirmed by customer
currency_code	string	Currency code
fare_summary	float	Fare summary including tolls, airport surcharges, cleaning fees, etc...

RESPONSE

```
{
    "result": "OK",
    "result_code": 200
}
```

ACCEPT PAYMENT

Confirmation of payment transaction. Confirmation of payment transaction, including optional tip decided by the customer.

RESTful	socket.io*
PUT /payment/:id	accept-payment

PARAMETERS

Name	Type	Description
token	string	Required access token

REQUEST

```
{
    "id": "12@tripthru.com",     // socket.io* only
    "confirmation": true,
    "tip": 1,
    "currency_code": "USD"
}
```

Name	Type	Description
confirmation	boolean	Boolean value to indicate if customer confirmed the payment
tip	float	Tip amount decided by customer (optional)
currency_code	string	Currency code

RESPONSE

```
{
    "result": "OK",
    "result_code": 200
}
```

AUTHENTICATION

OAUTH2.0 AUTHORIZATION

The OAuth2.0 implementation is based on Google implementation, which documentation can be seen in [1], so, this document follows the same structure in that page and further details can be collected there as well.

1. Register Application

An application must be manually registered in our system by our support team.

2. Obtain the API Key

This will be supplied in your account settings at www.tripthru.com

Once the application has the unique API Key, it is able to request authorization, doing a POST HTTPS request to the Auth URI: https://api.tripthru.com/oauth2/auth with the following parameters:

Parameter	Value/Type
key	API Key supplied by the Fleet
response_type	"code"
client_id	Network id
redirect_uri	URL for redirection (optional)

scope	Empty

The authorization request is stored in TripThru's database and the network is advised on its existence. The first response follows the standard below: When the request was already sent and it is sent again with no change from network side, the response above will be returned.

```
{
    "status": "OK",
    "status_code": 200,
    "authorization": "Waiting"
}
```

When the Network revokes the request, this is the response returned:

```
{
    "status": "OK",
    "status_code": 200,
    "authorization": "Revoked"
}
```

After the Network accepts the request from the settings page at www.tripthru.com, the **Authorization Code** is returned, like below:

```
{
    "status": "OK",
    "status_code": 200,
    "code": "50f6c144b6c1211721000002",
    "authorization": "Accepted"
}
```

3. Obtain tokens

In OAuth2.0, there are two important tokens available: access token and refresh token.

The access token is given for each session and expires every 15 days from the date obtained.

The refresh token only expires when application access is revoked; therefore, it does not automatically expire. This token is used by the application to request a refresh of the access token as it expires. This token allows an access token to be granted without requiring a new user authorization.

To request the tokens for the first time, send a POST request https://api.tirpthru.com/oauth2/token with the following parameters:

```
{
    "refresh_token": "PCLXSFJEsV9USwGs7xDNP5Sb5nJmkzFt",
    "access_token": "50dd7398b6c121557a000007",
    "token_type": "Bearer",
    "expires_in": 1247072
}
```

Parameter	Value/Type
code	Authorization code
client_id	Network id
client_secret	The client created upon registration
redirect_uri	URL for redirection (optional)
grant_type	“authorization_code”

Both “refreshtoken” and “accesstoken” must be stored. Access Token will be used for any API method call, while Refresh Token will be used only to request a new Access Token.

The token_type "Bearer" defines a simply way for checking accesstoken by passing it as a request parameter. The parameter **expiresin** counts how many seconds to expiration of the access_token.

To request a new Access Token, send a POST request https://api.tripthru.com/oauth2/token (same as above) with the following parameters:

```
{
    "access_token": "50dd7398b6c121557a000007",
    "token_type": "Bearer",
    "expires_in": 1246742
}
```

Parameter	Value/Type
refresh_token	The given refresh token
client_id	Network id
client_secret	The client created upon registration
grant_type	“refresh_token”

Note: if a new Access Token is requested prior to existing tokens expiration, the current token value will be returned. The API supports more than one active authorization at once, each with different tokens and sessions.

4. Send API method calls

All API methods must be called with a GET parameter in the following pattern: “...?accesstoken=ACCESSTOKEN_HERE”

Every method sent via POST or PUT must include its parameters as JSON object within the body, instead of the standard POST parameters.

5. Revoke access

You can revoke the current access token (without losing the whole authorization) by calling a POST request to https://api.tripthru.com/oauth2/revoke with the following parameters:

Parameter	Value/Type
client_id	Network id
client_secret	The client created upon registration
grant_type	“refresh_token”
refresh_token	Refresh token for checking
access_token	Access token to be revoked

JSON data response:

```
{
    "status": "OK",
    "status_code": 200
}
```

To revoke the whole authorization, you can call a POST request to https://api.tripthru.com/oauth2/revoke with the following parameters:

Parameter	Value/Type
client_id	Network id
client_secret	The client created upon registration
grant_type	“refresh_token”
refresh_token	Refresh token for checking

JSON data response:

```
{
    "status": "OK",
    "status_code": 200
}
```

VERIFY ACCESS TOKEN

Returns True if token and id are authorized. Checks if token is valid.

```
POST /verify
```

PARAMETERS

Name	Method	Type	Description
token	GET	string	Required access token
id	POST	string	Application client id

RESPONSE

If successful, this method returns a response body with the following structure:

```
{
    "status": "OK",
    "status_code": 200
}
```

API REFERENCES

TIMESTAMPS

Timestamps follow ISO RFC 3339 [1] standard DST "Z" (UTC time) to format date/time values,

Example:

```
2012-11-08T15:56:46Z
```

“Z” is not required to return UTC time values.

1. ISO RFC 3339 Timestamp specification
    - http://tools.ietf.org/html/rfc3339
2. ISO RFC 3339 Timestamp specification for timezones
    - http://tools.ietf.org/html/rfc3339#section-4.2

HTTPS STATUS CODES

Status Code	Description
200	OK. Everything worked as expected
400	Malformed request
401	Unauthorized the request requires user authentication
404	Not found
422	The request body is parse-able however with invalid content
430	Rejected
500	Internal Server Error

socket.io

To establish the socket.io connection include the access token in the connect query.

Example:

```
io.connect(url, {
    query: querystring.stringify({token:token}),
    transports: ['websocket']
});
```

Once the connection is established the requests are close to identical as the restful endpoint. Just include the {id} parameters inside the request body instead of the URL when using WebSockets.

USE CASE

DISPATCH AND TRACK EXAMPLE

Dispatch a trip and track it from start to finish

Originating network: GetTaxi

```
1 - GetTaxi dispatches a trip with no servicing network specified
POST /trip/3046@gettaxinewyork@tripthru.com
Request:
{
    "customer": {
        "name": "Benjamin",
        "local_id": "en_US",
        "phone_number": "233-553-2343"
    },
    "pickup_location": {
        "lat": 10.78107,
        "lng": 106.693294,
        "description": "will be waiting at main entrance."
    },
    "pickup_time": "2015-02-21T23:16:08.229Z",
    "dropoff_location": {
        "lat": 10.775779,
        "lng": 106.685451
    },
    "passengers": 1,
    "luggage": 0,
    "payment_method_code": "cash"
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

2 - TripThru broadcasts a quote request to all networks that serve the requested location

```
GET /quote/3046@gettaxinewyork@tripthru.com
Request sent to EasyTaxi and GrabTaxi:
{
    "pickup_location": {
        "lat": 10.78107,
        "lng": 106.693294
    },
    "pickup_time": "2015-02-21T23:16:08.229Z",
    "dropoff_location": {
        "lat": 10.775779,
        "lng": 106.685451
    },
    "payment_method_code": "account",
    "product_id": "taxi",
    "passengers": 1,
    "luggage": 0
}
EasyTaxi response:
{
    "quotes": [
        {
            "network": {
                "id": "easytaxi@tripthru.com",
                "name": "EasyTaxi"
            },
            "product": {
                "id": "easytaxi1@tripthru.com",
                "name": "EasyTaxi 1"
            },
            "eta": "2015-02-21T23:17:15.645Z",
            "fare": {
                "estimate": "12.5",
                "low_estimate": "11.5",
                "high_estimate": "13.0",
                "currency_code": "USD"
            },
            "duration": "875",
            "distance": "5.5"
        }
    ],
    "result": "OK",
    "result_code": 200
}
GrabTaxi response:
{
    "quotes": [
        {
            "network": {
                "id": "grabtaxi@tripthru.com",
                "name": "GrabTaxi"
            },
            "product": {
                "id": "grabtaxi1@tripthru.com",
                "name": "GrabTaxi1"
            },
            "eta": "2015-02-21T23: 29: 05.645Z",
            "fare": {
                "estimate": "11.5",
                "low_estimate": "11.5",
                "high_estimate": "13.0",
                "currency_code": "USD"
            },
            "duration": "875",
            "distance": "5.5"
        }
    ],
    "result": "OK",
    "result_code": 200
}
```

3 - TripThru selects EasyTaxi based on a better quote ETA and forwards the dispatch to EasyTaxi

```
POST /trip/3046@gettaxinewyork@tripthru.com
Request:
{
    "customer": {
        "name": "Benjamin",
        "local_id": "en_US",
        "phone_number": "233-553-2343"
    },
    "pickup_location": {
        "lat": 10.78107,
        "lng": 106.693294,
        "description": "will be waiting at main entrance."
    },
    "pickup_time": "2015-02-21T23:16:08.229Z",
    "dropoff_location": {
        "lat": 10.775779,
        "lng": 106.685451
    },
    "passengers": 1,
    "luggage": 0,
    "payment_method_code": "cash"
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

4 - EasyTaxi forwards accepted status update to TripThru

```
PUT /tripstatus/3046@gettaxinewyork@tripthru.com
Request:
{
    "status": "accepted",
    "eta": "2015-02-21T23:16:08.229Z",
    "driver": {
        "name": "Antonio",
        "location": {
            "lat": 10.78,
            "lng": 106.69
        }
    }
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

4.1 - TripThru forwards the update request to GetTaxi

5 - Driver is now en route, EasyTaxi sends an update status request

```
PUT /tripstatus/3046@gettaxinewyork@tripthru.com
Request:
{
    "status": "en_route",
    "eta": "2015-02-21T23:14:15.193Z",
    "driver": {
        "name": "Antonio",
        "location": {
            "lat": 10.78,
            "lng": 106.69
        }
    }
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

5.1 - TripThru forwards the update request to GetTaxi

6 - Driver picked up the customer, EasyTaxi sends an update status request

```
PUT /tripstatus/3046@gettaxinewyork@tripthru.com
Request:
{
    "status": "picked_up",
    "eta": "2015-02-21T23:17:17.422Z",
    "driver": {
        "name": "Antonio",
        "location": {
            "lat": 10.78107,
            "lng": 106.693294
        }
    }
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

6.1 - TripThru forwards the update request to GetTaxi

```
PUT /tripstatus/3046@gettaxinewyork@tripthru.com
Request:
{
    "status": "completed",
    "eta": "2015-02-21T23:17:17.422Z",
    "driver": {
        "name": "Antonio",
        "location": {
            "lat": 10.78107,
            "lng": 106.693294
        }
    }
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

7 - Driver reached drop off location, EasyTaxi sends an update status request

7.1 - TripThru forwards the update request to GetTaxi

```
POST /payment/3046@gettaxinewyork@tripthru.com
Request:
{
    "fare": 5.60,
    "currency_code": "USD"
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

8 - Trip is complete, customer is pending to confirm payment. EasyTaxi requests payment confirmation to TripThru

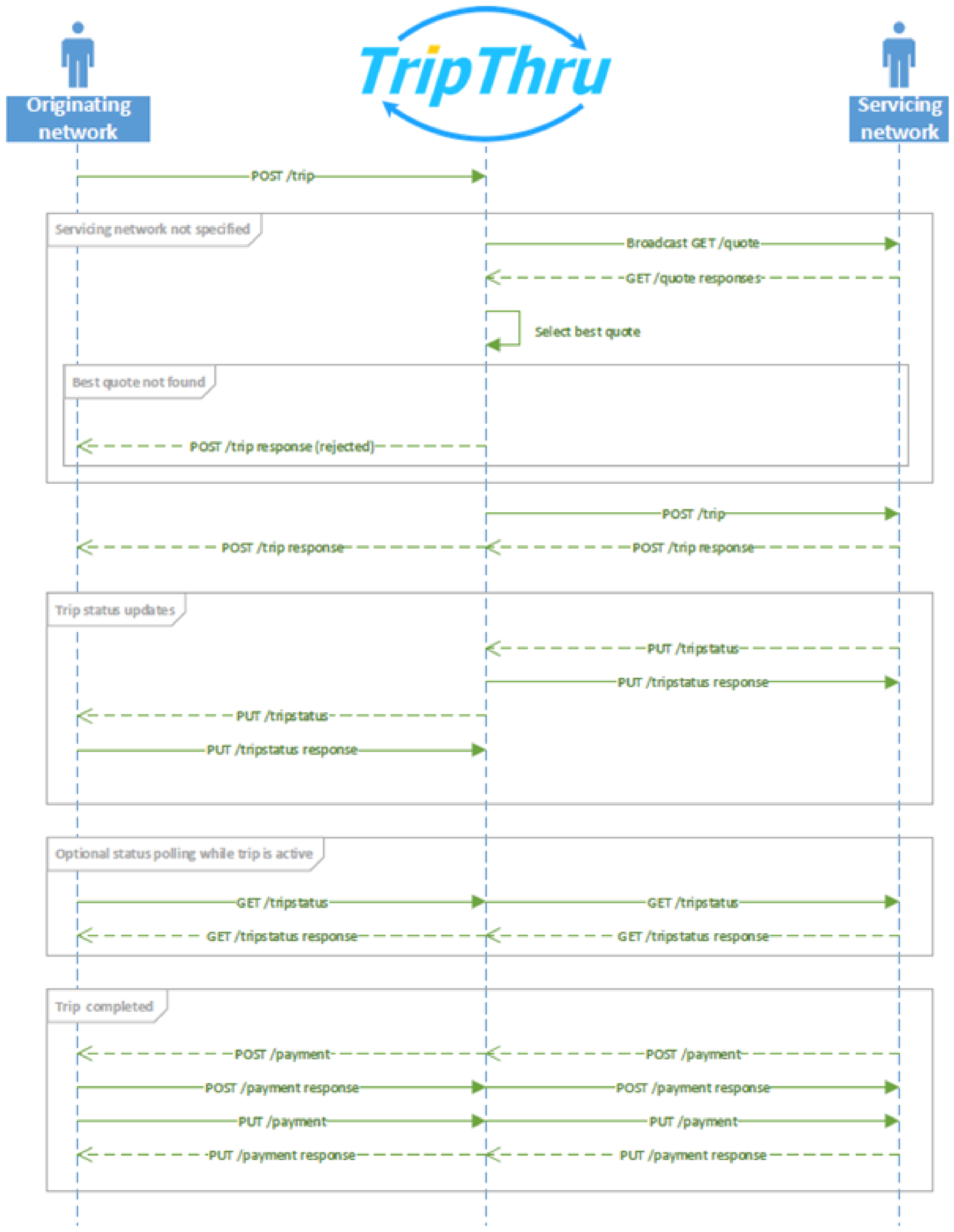
8.1 - TripThru forwards payment confirmation request to GetTaxi

8.2 Customer confirms payment, GetTaxi sends payment confirmation update request

```
PUT/payment/3046@gettaxinewyork@tripthru.com
Request:
{
    "confirmation": true,
    "tip": 1.00,
    "currency_code": "USD"
}
Response:
{
    "result": "OK",
    "result_code": 200
}
```

8.3 - TripThru forwards payment confirmation update request to EasyTaxi

TRIPTHRU COMMUNICATION FLOW DIAGRAM


Originating network
TripThru
Servicing network
POST /trip
Servicing network not specified
Broadcast GET /quote
GET /quote responses
Select best quote
Best quote not found
POST /trip response (rejected)
POST /trip
POST /trip response
POST /trip response
Trip status updates
PUT /tripstatus
PUT /tripstatus response
PUT /tripstatus
PUT /tripstatus response
Optional status polling while trip is active
GET /tripstatus
GET /tripstatus
GET /tripstatus response
GET /tripstatus response
Trip completed
POST /payment
POST /payment
POST /payment response
POST /payment response
PUT /payment
PUT /payment
PUT /payment response
PUT /payment response