Global Dispatcher - Use Case Catalogue

Document Control			
Version	**Date**	**Author**	**Changes**
0.1	27/11/2016	Jonathan Beebe	First Draft
0.2	08/12/2016	Jonathan Beebe	Added cancellation use case
0.3	18/01/2018	Jonathan Beebe	Further refinement
1.0	10/05/2018	Jonathan Beebe	Peer Review updates

Global Dispatcher - Use Case Catalogue

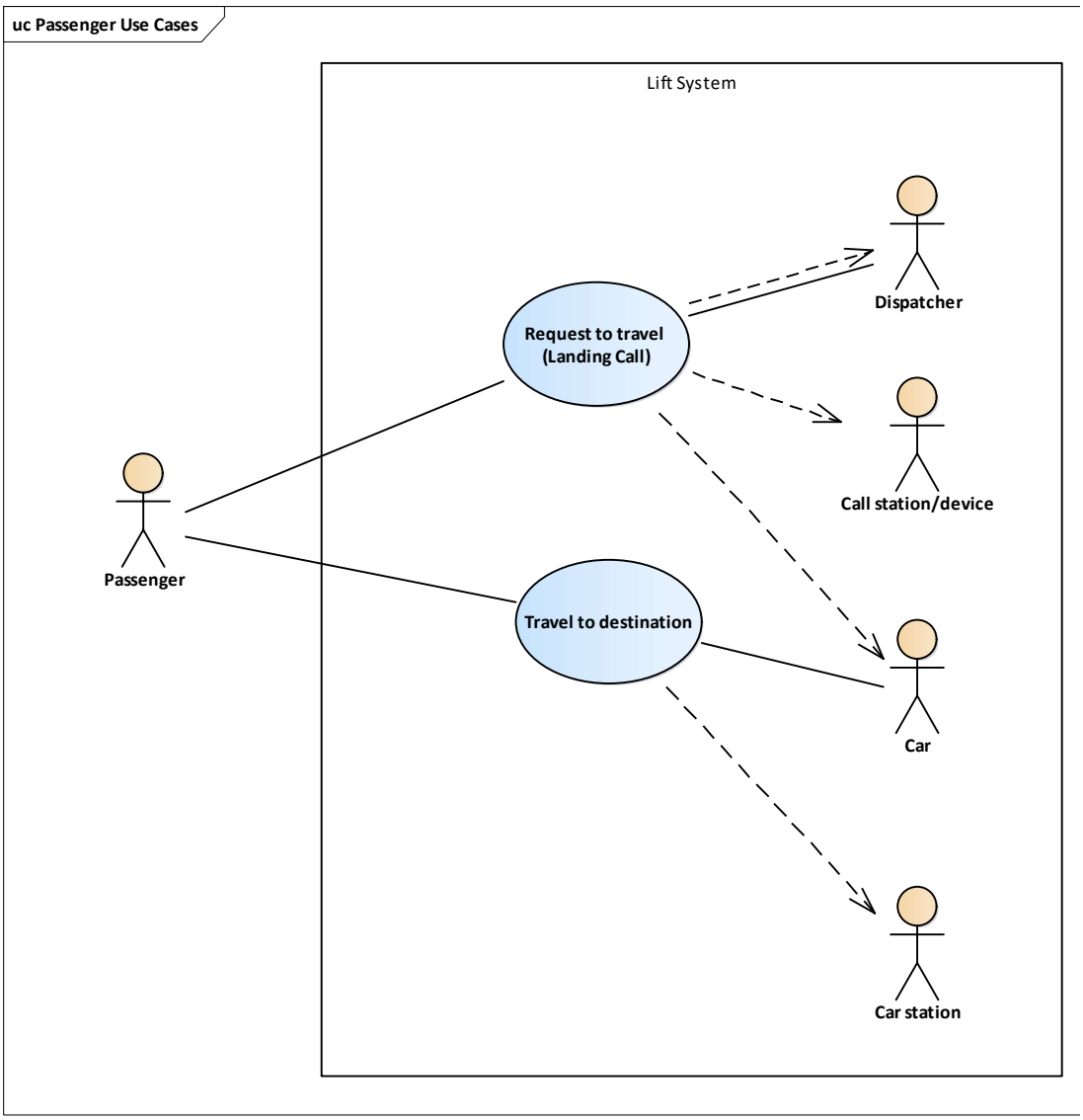
Introduction

This document contains the outlines of the key use cases of the Global Dispatcher Service. The use cases identify the interactions which must take place between the Global dispatcher service and its clients.

The use cases should be read purely as a statement of the required behaviour and do not have any bearing on the physical implementation or deployment of components of software. Where messages are defined as request and response pairs the purpose is to identify the information required to support the functionality and not the technology to be used to convey the messages nor the individual components that will generate/receive them - i.e. a request might simply be a function call inside a single .dll or it might be web service request over the Internet. This detail will be optimised during the detailed design phase.

The elements of information which are referenced in the document (usually with CamelCase names relate to the Standard Elevator Information Schema) which is published at www.std4lift.info then follow the "Schema Documentation" link in the left-hand frame. (*It is planned to incorporate this schema into the same model that contains the use cases*).

Passenger Use Cases



The Passenger use cases describe the flow of interactions between the passenger and the elements of the lift system to achieve the use case objective. They are a view of the interaction of the system with its external environment.
The main actor of these use cases is "the passenger" but it is important to note that many instances of these use cases may be active concurrently and asynchronously so they may exist in as many different stages of completion. One passenger, having partly completed one of the above Use Cases may affect the experience of another passenger or be affected by another passenger who is at a different stage of completion of the same or a different

Use Case. This means that other passengers are referenced in the use case text who are not the main actor and should not be considered as actors in the use case in any sense.

Request to travel (Landing Call)

Passenger approaches a call station or activates an application on a smart device.
Passenger registers a request to travel which includes:

- Origin floor
- Destination direction OR destination floor

Optionally other information may be supplied in the request such as:

- Priority status of the passenger or requested journey
- Number of passengers travelling as a group
- Access security details

The dispatcher responds via the call station (or smart device):

- Confirms the request (e.g. illuminated button)

plus optionally:

- Confirms the car that the passenger will travel in

The dispatcher also communicates the assignment directly to the assigned car.
The car responds by adding the assigned call to its travel plan and in due course departs with a destination of the passenger's origin floor.

On arrival at the passenger's origin floor the car cancels the passenger's call and informs the dispatcher of the call cancellation.

The dispatcher indicates the cancellation to the passenger via the call station and optionally other indicator devices (including possibly the smart device, if in use).
The dispatcher records the System Response Time for performance analysis purposes.

If the dispatcher has not been informed of the passenger's destination floor the passenger's call is then deleted from the list of current calls. However, if the destination floor is known then the call is retained but its status is changed to "Answered".

Travel to destination

The car arrives at the passenger's origin floor and opens its doors to allow
any arriving passengers to exit from the car
the waiting passengers to enter the car

If the dispatcher already knows the passenger's destination floor it will establish with the arriving car a car call for the destination floor, which may be displayed on the car station.

If the dispatcher does not know the passenger's destination floor, then the passenger registers their destination floor via the car station.

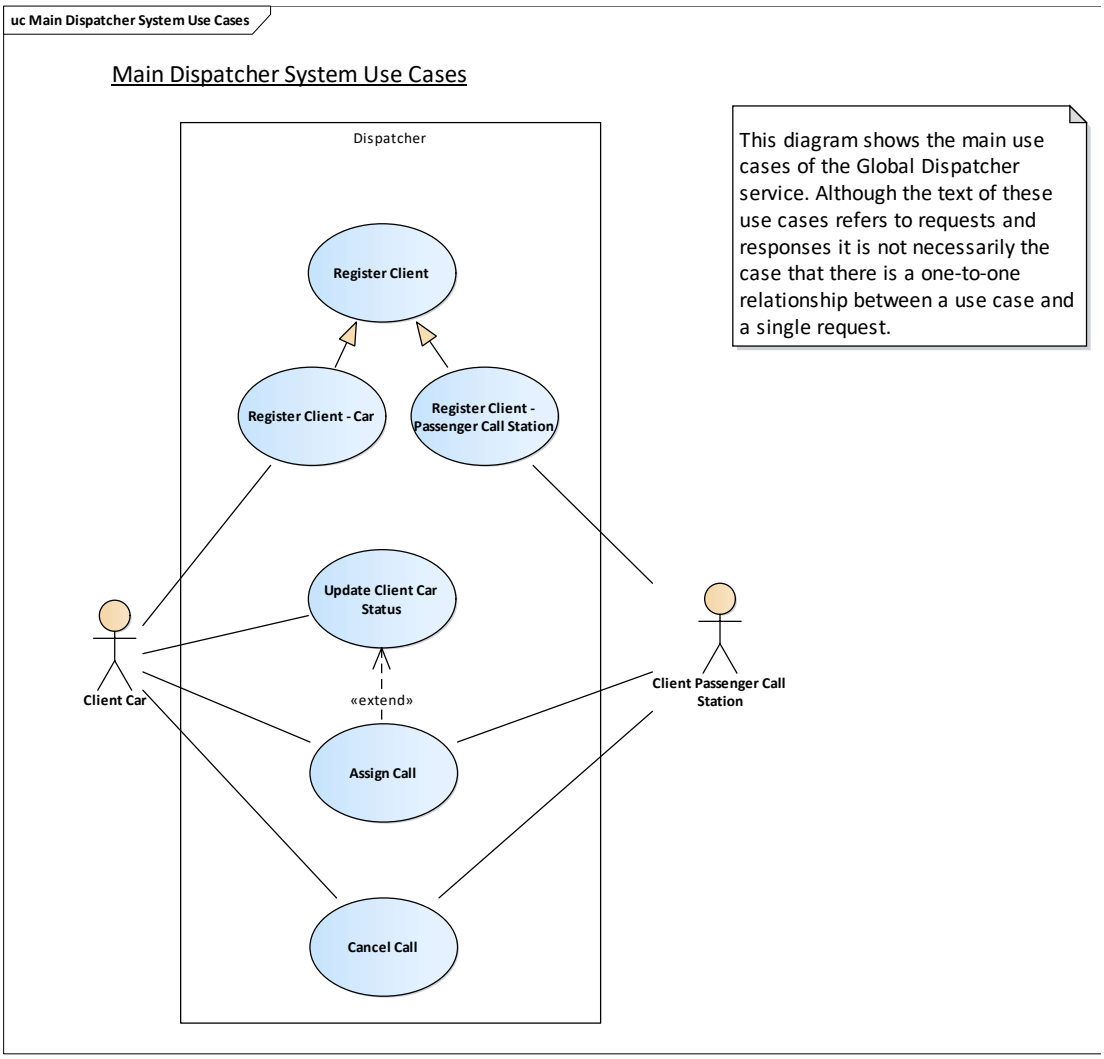
Whichever mechanism is used to register the car call results in a stop at the passenger's destination floor being included in the car's travel plan (although this may already be in place due to the requests of other travelling or waiting passengers).

After allowing passengers to transfer to and from the current stop floor, the car closes its doors and departs for the next destination in its travel plan.

On arriving at the passenger's destination floor the car cancels the car call for that floor and informs the dispatcher.

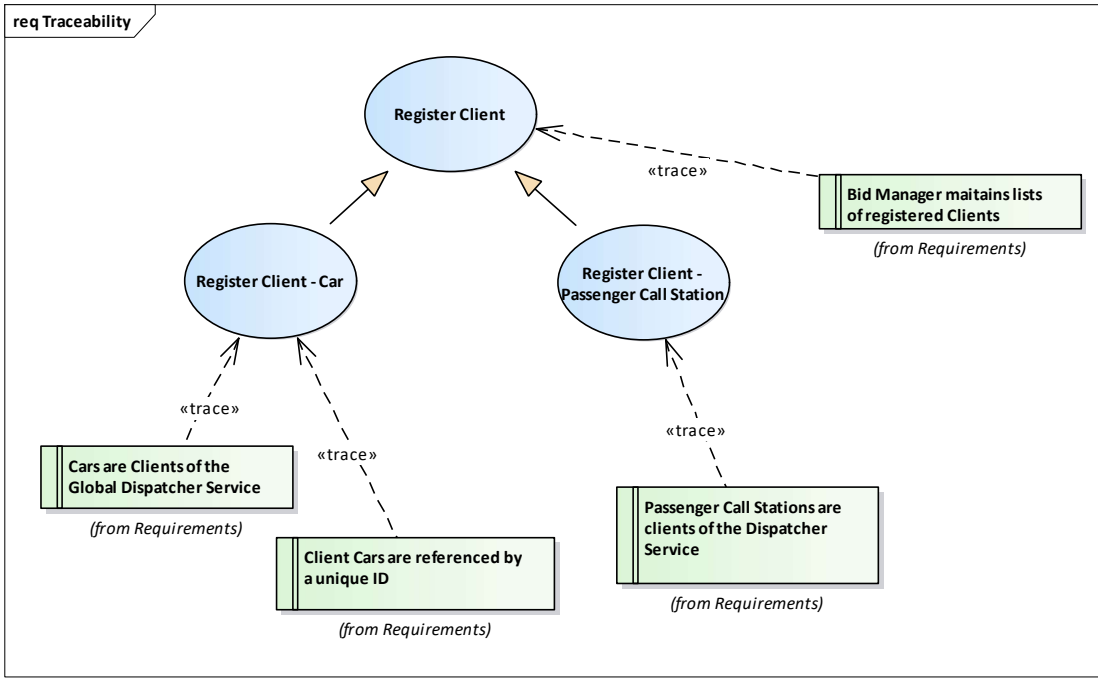
The car opens its doors to allow the arriving passenger to depart and then any further waiting passengers to enter the car to begin travel to their own destination floors.

Main Dispatcher System Use Cases



The System use cases describe the flow of interactions between the elements of the lift system in order to support the fulfilment of the Passenger use cases. In that sense they are a simple view onto the internal operation of the system.
(*N.B.there is not a simple relationship between the Passenger and System use cases*).
Communication/interaction between asynchronous processes is well suited to the Event Driven Pattern of programming (Richards, 2015) - examples change of next possible stopping floor, call registration, etc.

Register Client



Register Client - CarA client car registers itself with the dispatcher by providing details of its static information:

- The client's own identifier
- Speed profile
- Door profile
- Reference Energy profile
- Floors served
- Rated Capacity
- Number of decks

This request for service implies that the client is interested in providing a transport service to passengers.

The dispatcher service responds by acknowledging the request and supplies an identifier in its response so that future interactions may be associated with the correct client.

Register Client - Passenger Call Station

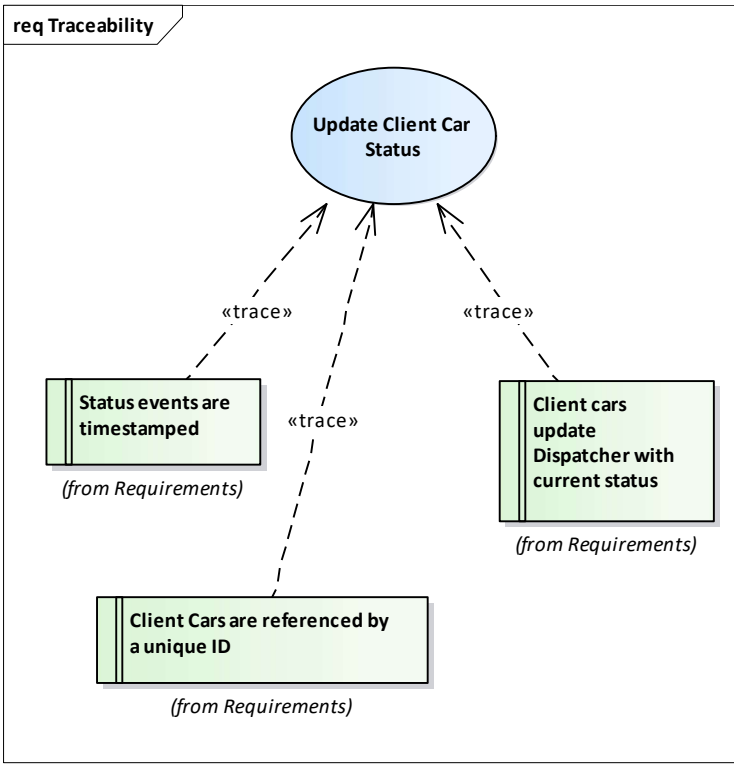
A client Passenger Call Station registers itself with the dispatcher by providing details of its static information:

- The client's own identifier
- Device type
- Floors served
- Location

This request for service implies that the client is able to request transport for passengers.

The dispatcher service responds by acknowledging the request and supplies an identifier in its response so that future interactions may be associated with the correct client.

Update Client Car Status



A client car announces a change of its state to the dispatcher by providing details of its dynamic information at the time they occur:

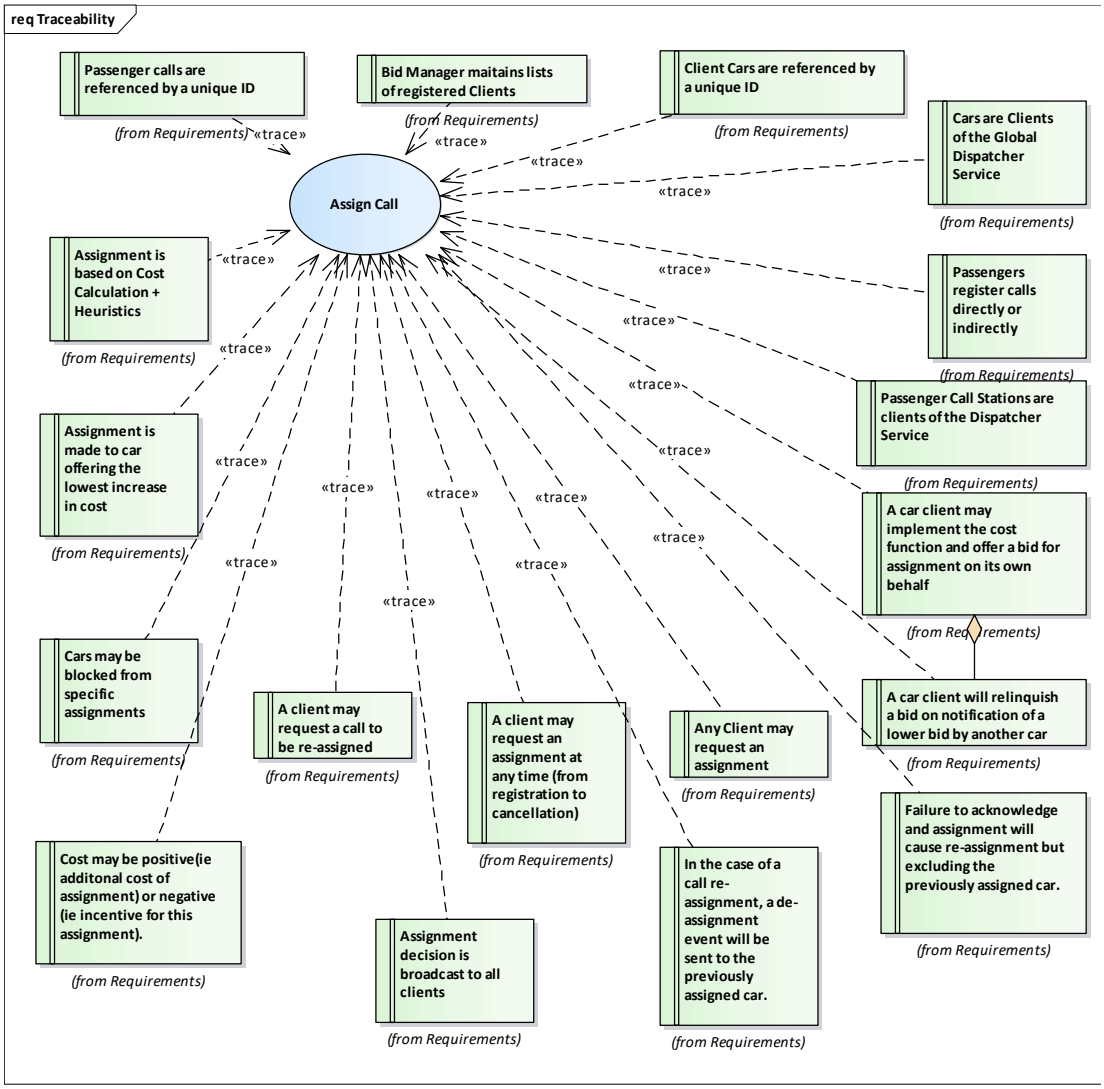
- The ID supplied by the Service in response to the Introduce Client request
- The client's own identifier
- Current Floor

- Direction
- Drive state
- Door state
- Travel Plan
- Shaft
- Currently registered calls (Car, Landing, Park)

All this information may be supplied in a single request (CarDynamicData) or, preferably, a sequence of incremental requests as each change of state event occurs (LogEventType). All dynamic data and events are time-stamped to avoid errors due to communication delays.

At the discretion of the specific dispatcher algorithm (not the dispatcher interface) an update of car status may initiate the re-assignment of calls, which is shown by the extends relationship to the Assign Call use case.

Assign Call



Any client (including passenger signalling device) may request a Call Assignment. The request includes the following information:

- Call Floor
- Direction or Destination Floor
- Registration time

in the form of a CallRegistration event.

N.B. *inclusion of Registration Time means this request may (re)occur at any time after the initial call registration event and may therefore be a request for a call to be re-assigned or simply a delayed request if no car was available for assignment earlier. In the case of re-assignment the dispatcher will have a record of the currently assigned car and this information may influence the result of the new assignment, though this would be*

a characteristic of the specific dispatcher algorithm and not of the dispatcher interface.

At the discretion of the specific dispatcher algorithm (not the dispatcher interface) an Assign Call request may initiate the re-assignment of other previously assigned calls, which will result in multiple executions of the Assign Call use case.

Main Flow

The dispatcher calculates the cost of assigning the call to each of the registered and available client cars according to its own internal algorithm design. The term "cost" is not restricted to a purely financial cost and may be evaluated in terms of one or more criteria such as

- waiting time,
- system response time,
- energy consumed,
- etc

as a function of the increase in the value of that parameter after the call has been assigned compared to the cost before it was assigned to the car. The algorithm may include penalties or incentives that are derived from a logical analysis which will modify the simple cost.

If an overriding criterion is included in the algorithm such as never assigning a call that would cause the car to become overloaded then that car will be marked as "blocked".

N.B. *Availability for assignment may be determined by a variety of properties such as:*

- *the operating mode of the car*
- *whether the car is able to service the call floor(s)*

However, these availability decision is a characteristic of the specific dispatcher algorithm and not of the dispatcher interface.

The response is broadcast via the dispatcher interface to all registered clients and includes (but is not limited to):

- the CallRegistration call event updated with the cost of assignment, where cost analysis is made in terms of the cost-function that is specific to the algorithm used by the dispatcher

plus optionally:

- a CallAssignment event where the Assigned To element is populated with the minimum-cost assignment details.
- a TravelPlan if the dispatcher is configured to hide assignments so that the car will by-pass an assigned call until it becomes the next landing call for the car.
- if the call was previously assigned to a different car the response will also include a Call Deassignment event.

The Assignment is only considered to be complete after a positive acknowledgement has been received from the dispatcher responsible for the assigned car.

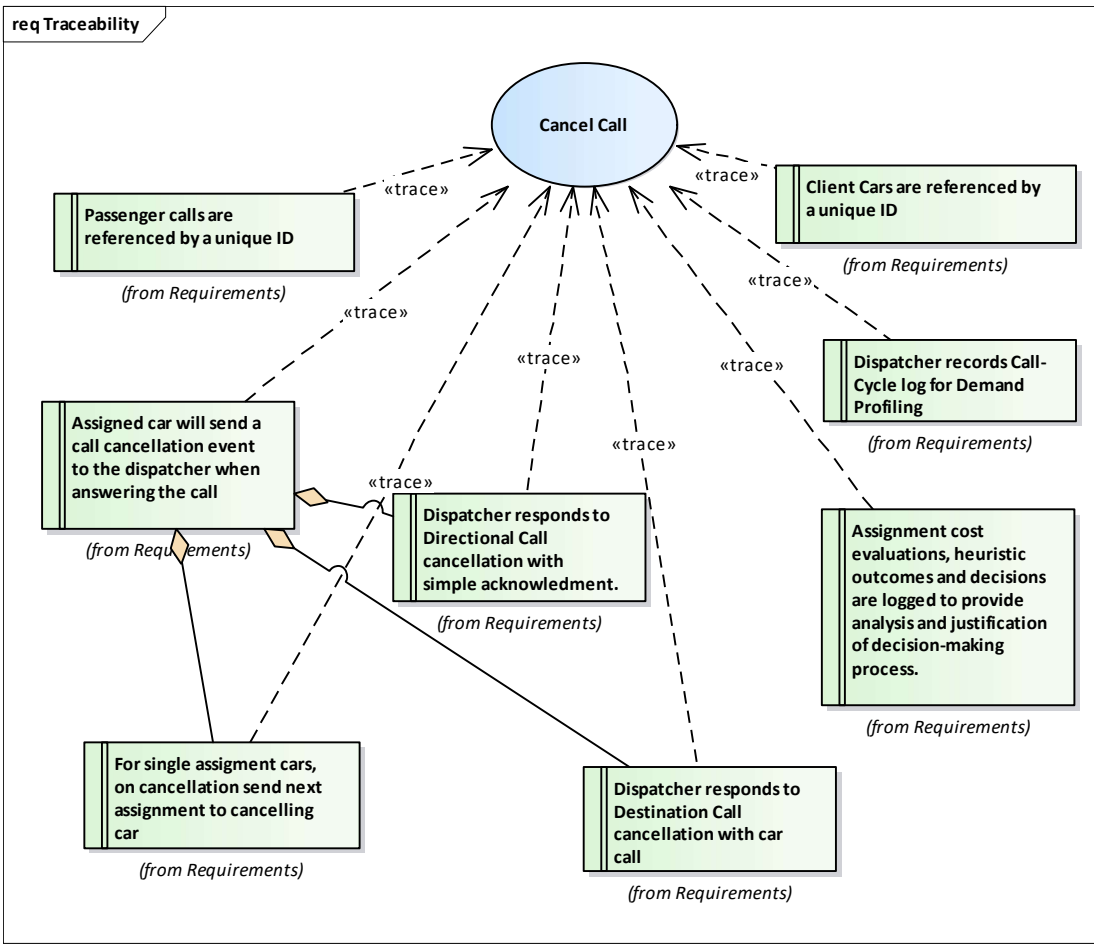
Alternative Path 1)

If no acknowledgement or a negative acknowledgement is received from an assigned client car then the call assignment will be repeated with that car excluded.

Alternative Path 2)

The dispatcher may be distributed as a number of collaborating instances, each responsible for a unique set of one or more registered client cars. In this case each dispatcher instance will respond with a "bid" for the minimum cost of assignment for adding the call to the travel plan of one of its registered client cars. If having bid, the dispatcher instance is notified by another instance of a lower bid, the assignment will be relinquished in favour of the lower cost bid.
In this mode, the Assignment is only considered to be complete after a positive acknowledgement has been received from the dispatcher responsible for the assigned car.

Cancel Call



When a car with an assigned LandingCall reaches the arrival Floor with the committed departure direction of its assigned LandingCall and starts to open its doors, the car sends a CallCancellation event (including the call ID) to the dispatcher.

The dispatcher matches the event in its current RegisteredCalls GroupDynamicData. If

the assigned call is a "destination call" the dispatcher responds to the car with a CarCall CallRegistrationEvent for the Destination floor of the assigned call. If the assigned call is a "directional call", the dispatcher simply acknowledges the cancellation.

If the dispatcher is configured for single Landing Call assignment it will send a CallAssignment for the next assigned landing call to the car, if one exists.

The dispatcher records a CallCycle LogEventType in an activity log (*Afurther use case describe how this activity log is queried by external clients)* . It may optionally also use the CallCycle as an input to building a DemandProfile LogEventType. DemandProfile LogEventTypes are then available for internal use by the dispatcher in applying its algorithm and may also be made available to external clients.