**How does Navigation Module work in TN7.0**

[Part1. Route Planning 1](#_Toc293925710)

[1. Does it always show? 1](#_Toc293925711)

[2. The basic flow 2](#_Toc293925712)

[3. Some principles 2](#_Toc293925713)

[4. Integration with OPENGL MAP engine. 2](#_Toc293925714)

[Part2. Moving Map 3](#_Toc293925715)

[1. Does it always show? 3](#_Toc293925716)

[2. The basic flow 3](#_Toc293925717)

[3. Rendering mode & interaction mode 3](#_Toc293925718)

[4. Information to display. 4](#_Toc293925719)

[5. Audio Guidance 4](#_Toc293925720)

[6. Traffic Alert 4](#_Toc293925721)

[7. GPS 5](#_Toc293925722)

[8. JuctionView 5](#_Toc293925723)

[9. Deviation 5](#_Toc293925724)

[10. ADI 6](#_Toc293925725)

[11. Control buttons on Navigation Screen 6](#_Toc293925726)

[12. End Trip 7](#_Toc293925727)

[13. Detour 7](#_Toc293925728)

[Part3. Turn Map 7](#_Toc293925729)

[1. Route Edges 7](#_Toc293925730)

[2. Audios 8](#_Toc293925731)

[3. Reload route edges 8](#_Toc293925732)

[Part4. Summaries 8](#_Toc293925733)

[1. Route summary 8](#_Toc293925734)

[2. Map summary 8](#_Toc293925735)

[3. Traffic summary 8](#_Toc293925736)

[4. Summaries switch control. 9](#_Toc293925737)

Part1. Route Planning

1. Does it always show?

The answer is NO. It’s a feature that can be controlled by upsell info (similar with server driven parameter).   
See FeaturesManager. FEATURE\_CODE\_ROUTE\_PLANNING\_MULTI\_ROUTE.  
If it is set to disabled, then we will see the setting screen instead of route planning screen.

1. The basic flow

If route planning is not disabled:

1. Put Origin/Destination into model (In the structure of Address).  
   If one is current location, create an Address with the lat/lon.
2. If one of Origin/Destination is current location, try to get current location data.  
   Once current location is retrieved, start to get route choices.  
   **Note: If it’s cell location, we set the Accuracy to the max value (50) that the server can accept and treat it as valid GPS data.**
3. After route choices coming, update view to show them on the map.
4. User can click edit route button to change options in setting screen.
5. User will click “navigate” or “directions” to start navigation/directions.  
   **Note: if user clicks navigate, we will check his account to see whether this feature is available. If not, we will show up sell page. (see ACTION\_CHECK\_ACCOUNT\_STATUS for details)**
6. If No GPS error or network error occurs before route choices being retrieved, if there is popup, will go to setting page if we close the popup.
7. No route, No GPS: show destination in the center of the map as the common map view.  
   No route, Origin/Destination available: show region that can contain them.  
   Have routes: show region that can contain the routes.
8. Some principles
9. We should use Address object within this module or with other module, not Stop or POI.
10. If we have got a location data to request route choices, we should always display current location at that point and **NEVER** update it on the map. This is requirement of PM.
11. Route Planning module need to save the location data that is used to get route choices, and pass it to Moving Map module. If it’s a cell location, we will trigger special logic on the start of Moving Map module.
12. Integration with OPENGL MAP engine.

This module is a typical one on using the map engine API.  
It shows how to add multi-routes on map and change the order and style.  
It’s also a good example of showing a specific lat/lon area on the specific part of the map view.  
See **updatePlanningMap()** for details.  
  
**Note:  
1).Route can be reordered.  
2).To change route style we must remove and add routes again.  
The route style is set in config\_default.json. If you want to change it, please contact people in US side.**

Part2. Moving Map

1. Does it always show?

The answer is NO. It’s a feature that can be controlled by upsell info (similar with server driven parameter).   
See FeaturesManager.FEATURE\_CODE\_DYNAMIC\_ROUTE.  
If is set to disabled, then we will see the up sell page after click “navigate”.

1. The basic flow
2. Enter Moving Map screen anyway.
3. Retrieve GPS to get route.   
   Special logic: if route planning module get route choices by cell location, just copy 3 of it to get route. In this case, show Grey car (means no GPS) and play no GPS audio at once. If GPS recovered, play “has GPS” audio and change car to normal at once.
4. Get route.
5. Start Navigation.
6. Rendering mode & interaction mode

By using OpenGl Map engine, you will find there’s lots of logic causing “rendering mode” & “interaction mode” changes. What’s them for?

**Rendering modes we use:**  
**RENDERING\_MODE\_3D\_NORTH\_UP** //3D, geography north is consist with screen north.

**RENDERING\_MODE\_3D\_HEADING\_UP** //3D, the direction of the engine is consist with screen north.

**RENDERING\_MODE\_2D\_NORTH\_UP** //2D, geography north is consist with screen north.

**RENDERING\_MODE\_2D\_HEADING\_UP** //2D, the direction of the engine is consist with screen north.

**Interaction mode we use:**

**Follow Vehicle** means the map will move automatically with the car. It’s the normal mode for Navigation.  
**Pan And Zoom** means the map won’t move until you pan it. It’s the mode for map viewing.

1. Information to display.

The information displayed and updated during navigation such as street name, turn type, dist to turn, dist to destination, ETA, etc, is got from **NavInfoEvent** which passed to Moving Map module by Nav Engine.

1. Audio Guidance

Every time we handle NavInfoEvent(about 1 time per second), DynamicNavInfoRequester will send request to get edges and dynamic audios. Nav engine will give moving map module NavAudioEvent to play audio. We use AudioComposer to compose the play list, and pass the list to AudioPlayer to get it heard.

1. Traffic Alert
2. Where is it from?  
   When navigation starts, TrafficAlertEngine will be started. It will start TrafficAlertJob, which will keep running during navigation. This job will send request to get new ones about every 5 minutes, and check current retrieved traffic alerts status and update UI accordingly each loop round. Moving Map module then update UI by the callback of TrafficAlertJob.
3. How TrafficAlertJob work?  
   After server returned current traffic incidents, TrafficAlertJob will try to process them:  
   **First**, tell them into two parts: traffic camera & speed trap, and other incidents. Both the old incidents and the new got incidents will be separated into two types. Now we have four Vectors: Vector A(old traffic camera/speed trap) Vector B(old other incidents) Vector C(new traffic camera/speed trap) Vector D(new other incidents).  
   **Second**, merge the old and new incidents. Vector A and C are merged one by one. Vector B and D are merged by the distance to current location on route. Now we have two vectors (E and F).  
   **Third,** merge E and F into G. E contains all the traffic cameras & speed traps, while F contains all other incidents. Add E first then F. G is such a vector: first part are all traffic camera & speed traps, following are other incidents sorted by distance.  
     
   After above process, it will find a “group” of incidents to be used to be displayed. How to define the group?   
   **First**, get the first element in Vector G. Let its distance to the car be firstDist.  
   **Second**, loop incidents in G. If the distance is in a range compare to firstDist, add it to the group.  
   **Third**, loop all the incidents in the group. Find the most severe one.  
   Fourth, tell moving map to display it and play its audio.
4. Click on the traffic alert will show the detail messages and provide the option to avoid it.
5. GPS
6. The car position is real GPS?  
   We usually don’t doubt about it but actually the answer is NO.   
   We will provide GPS to Nav Engine but, it will calculate where you should be **ON THE ROUTE**. That means, the location we use to update car position is calculated by Nav Engine. That’s why we always see car on the route except ADI mode.
7. How do we play the audio “NO GPS”?  
   Nav Engine will give us NavGpsEvent. From there we will know the satellite account and if it’s below 0, we will say “you don’t have GPS signal…..” , and the car & bottom part become grey. Next time if GPS recover, we will say “you now have GPS signal…” and change the car and bottom part into normal. Of cause each change need a period of stable status, in our code about 30s.
8. JuctionView

What is Juction View:  
When the car runs into one turn, the camera declination will become higher. After the car exit the turn, the camera declination will recover to normal.

See **enterJunctionView**() and **exitJunctionView()** for detail.

1. Deviation
2. When will deviation happen?  
   As we know in “GPS” part, the car position is calculated by Nav Engine, and it’s always on route. That means it allows our real position not on the route in some range. But if it goes out of the range, the Nav Engine will trigger special logic:  
   If there are some other routes near the car, will trigger logic of deviation. Client will send a request to backend to get a new route based on current car position.  
   If there’s no route near the car in some range, then it’s different case. ADI will be triggered and try to drag user back to current route.
3. What will we do in deviation?  
   Send a request to backend(see **handleDeviation()** ).
4. What will happen if we are in other page when deviation occurs?  
   If we are not in moving map page when deviation occurs, such as summaries, search along, etc, we will force back to moving map page and show “rerouting”. But there are some exceptions:  
   i. traffic summary🡪 avoid segment   
   ii. Traffic summary 🡪 minimize all delays  
   iii. Moving map 🡪 avoid traffic alert  
   It’s not hard to understand why they are different: each of them will trigger a request to get a new route based on current location. There is no need to interrupt this request and trigger a new request.  
   (see **setDeferDeviation(boolean deferDeviation)** for details)
5. ADI
6. When will deviation happen?  
   As discussed in Deviation, if we deviate from the route but around you there’s no routes that are so near, ADI will be triggered to drag you back to original route.
7. What does ADI look like?  
   The map becomes 2D. The zoom level will become higher. A red line will connect the car and the “enter point” of the route.  
   By doing these we want to make sure user can see the route around him clearly.
8. Where does it come from?  
   Nav Engine will use the map tiles we got to judge ADI. MapTileAdiStatusJudger is the tool to do this. The detail is hard to describe here and you can contact the experts to learn about it.
9. Where do we get map tiles?  
   One question about this is the map tiles. Currently we use protobuf to do requests and get the responses separately. We can’t find a method that send the map tile request separately. Then where do we do this? The answer is that it’s bundled with some other requests. Such as the request to get route, it will return tiles around the origin point to judge ADI at the beginning; we will also get tiles in the request of “Get\_Extra\_Edge”, which is always sent during navigation, so that we can get tiles all along the route.
10. Control buttons on Navigation Screen
11. Traffic button (popup on left top, can be seen after tapping on screen)  
    Will go to traffic summary.
12. Zoom buttons (popup on right top, can be seen after tapping on screen)  
    Will zoom in/out map.
13. Compass (on the left bottom of moving map screen.)  
    Do you know it can be clicked? It can trigger 2D/3D change. Try it!
14. Navigation info top container.  
    The top part of the screen. It can be clicked also~! Will replay previous audio guidance.
15. Traffic alert popup  
    Show traffic incidents ahead. Clicking it will see detail message and provide option to avoid it.
16. Back to Nav button  
    If you pan on Moving Map, the mode will be changed from **Follow Vehicle** to **Pan And Zoom.**By press this button you can continue to see the normal navigation map moving with car.
17. End Trip

When near the destination, Nav Engine will tell listener about it. Moving map view enter end trip mode.

In this mode we stop to update the screen (it’s a transient state). But the car position still need to be updated. So we start MapVehiclePositionService to update car.

If user continues to drive pass destination or deviates around destination, new route will be generated. New navigation starts.

1. Detour
2. What is detour?

While you are in moving map mode, do search along and drive to the POI you want.

After you reached the POI, or cancelled it during driving toward it, detour happens.

For general navigation process, if you want to end trip, it will ask you “whether you want to end the trip” will selection “yes” and “no”.

But for detour, you have 3 selections: “resume previous trip” “end trip” and “cancel”.

That means you can back to previous tour easily.

1. How does detour work?

TripsDao will remember the last trip and the detour trip. When navigation occurred, check last stop to decide whether it’s detour.

Part3. Turn Map

1. Route Edges

When we entered turn map, maybe only a part of edges are downloaded. In this case, the route drawn on the map is not complete. But user can enter each turn by clicking one segment on route summary. What if no route data for that segment?

TurnMapWrapper is the man to consider about it. When we enter a segment, it will check whether the route edge data presents for the segments around selected one (backward 3, and forward 4 segments). If not, play progress bar and it will request route edges for these segments.

1. Audios

Turn map model will register into Notifier to do the audio request continuously.

1. Reload route edges

After new route edges come, you must reload the route in map engine to make it shown. Otherwise the route displays the same as before.

Part4. Summaries

1. Route summary

Display the segments as a table. The only difference with previous version is that the car icon will automatically move down during navigation! This is done by registering listener into the NavEngine, and getting the NavInfoEvent which contains current segment message.

1. Map summary

Map summary display the whole route in a high zoom level. As we know in navigation part, the route is downloaded part by part. If the route is downloaded fully when the navigation starts, then just add the route points into map engine to show it. But if it’s a long route (to New York), obviously we can only display a short part of the whole tour.

DecimatedRoute is used to resolve such problem. It contains only one segment, and the points in it are loose points. This means it is not precise, but just a summary of the trip.

1. Traffic summary

Traffic summary displays traffic message of each segment in a table.

You can view traffic detail, avoid segment forward, or minimize all delay of the trip.

1. Summaries switch control.

You may noticed that there is SummaryControlController. Really strange name…

What is it for?

Summaries can jump between each other freely. It is ideally to make them child of moving map/ turn map. When switch happen, current summary controller post controller event and moving map/ turn map start another summary.

It’s very clear. But we met one problem: when entering map summary and traffic summary, there may be network request with progress bar shown.

For example, we want to jump from route summary to traffic summary. You can see a progress box (shown by traffic summary controller because it’s sending request) lying above route summary. If user cancelled this request, what will happen?

Route summary has already been released after posting controller event.

Traffic summary will be released after request being cancelled.

So it will go back to traffic summary’s super controller: moving map controller.

But it’s strange that you cancelled a progress box upon route summary screen (you can still see it below the progress box), but it jump back to moving map.

So we need to have a control class who will remember the last activate summary module, and reactivate it in this case.