259167

Contents

1 259167

2 259167 - SS-7MFLW3/ NAVTEQ

3 FUNCTIONAL OVERVIEW

3.1 Client Requirement

Provision of new navteq mapping solution to be applied to MTS databases.

3.2 Solution

Keep all of the existing code within MTS for generating longitude/latitude and route distance/time requests (status is QUEUED) along with the code to update the relevant location/trip data once processed by Map Point (status moved from PROCESSED to APPLIED).

Replace Map Point with a new D/B job to pick the QUEUED requests, send an XML call to NAVTEQ (either Lat/Long or Route), interpret the results and update the appropriate data in the request moving it onto to status PROCESSED.

3.3 Scope

This change will be applied to system version 10.5.0 on CONTST and once approved it can be expanded to all test systems and then moved into production after final testing on each test system.

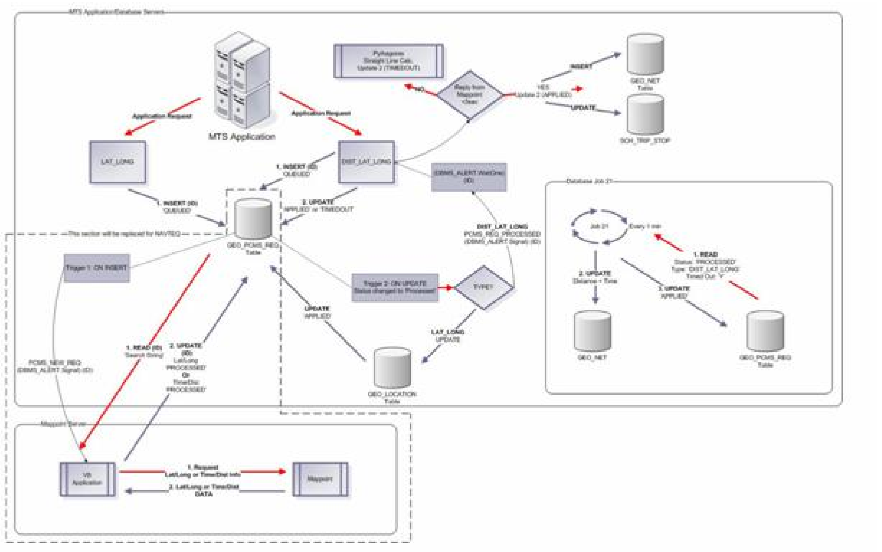
4 SET-UP

4.1 Pre-Requisites

System is already coded to generate long/lat and route requests (status QUEUED) and once processed by Map Point (status moved to PROCESSED) then they will already be used to update the relent data within MTS (moving the requests to status APPLIED).

5 FUNCTIONAL DESCRIPTION

The current process flow for requesting information from MTS to Map Point is as follows :-


MTS Application/Database Server
MTS Application
Application Request
Application Request
LAT_LONG
DIST_LAT_LONG
Pythagoras Straight Line Calc. Update 2 (TIMEDOUT)
Reply from Mappoint <Time
NO
YES Update 2 (APPLIED)
INSERT
UPDATE
GEO_NET Table
SCH_TRIP_STOP
1. INSERT (ID) 'QUEUED'
1. INSERT (ID) 'QUEUED'
2. UPDATE 'APPLIED' or 'TIMEDOUT'
DBMS_ALERT.WaitOne (ID)
GEO_PCMS_REQ Table
This section will be replaced for NAVTEQ
Trigger 1: ON INSERT
DIST_LAT_LONG PCMS_REQ_PROCESSED (DBMS_ALERT Signal) (ID)
Trigger 2: ON UPDATE Status changed to 'Processed'
TYPE?
UPDATE APPLIED
LAT_LONG UPDATE
GEO_LOCATION Table
PCMS_NEW_REQ (DBMS_ALERT Signal) (ID)
1. READ (ID) 'Search String'
2. UPDATE (ID) Lat/Long PROCESSED' Or Time/Dist PROCESSED'
Database Job 21
Job 21
Every 1 min
1. READ Status: 'PROCESSED' Type: 'DIST_LAT_LONG' Timed Out: 'Y'
2. UPDATE Distance + Time
3. UPDATE 'APPLIED'
GEO_NET
GEO_PCMS_REQ Table
Mappoint Server
VB Application
1. Request Lat/Long or Time/Dist Info
2. Lat/Long or Time/Dist DATA
Mappoint

Code already exists within MTS which generates Map Point requests for information onto a table called GEO_PCMS_REQ (REQ_STATUS = ?QUEUED?) for either longitude/latitude information for a location (REQUEST_TYPE = ?LATLONG?) or for distance/time information between trip stops (REQUEST_TYPE = ?DIST_LATLONG?).

This code will be left unchanged and the requests for information generated as now.

NB) A trigger (T_GEO_PCMS_REQ_ID) on the creation of the ?QUEUED? requests exists which currently generates a DBMS_ALERT.SIGNAL of type ?PCMS_NEW_REQ? for the request ID which is currently picked up by the Map Point VB program.

The requests are currently being processed by Map Point adding the additional information onto the request table (both LONGITUDE and LATITUDE for a location or TOTAL_KM and TOTAL_MINS for a route). It also moves the request onto REQ_STATUS = ?PROCESSED?.

It is this process that is going to be replaced by the new NAVTEQ code (see later).

NB) A trigger (T_GEO_PCMS_REQ_PROC_SIGNAL) on the changing of a request to status ?PROCESSED? currently exists which issues a DBMS_ALERT.SIGNAL of type ?PCMS_REQ_PROCESSED_? <Request ID>. This alert is still required as this tells the trip planning screen that the time/distance calculation has completed so that they can then be used to update both the network table (GEO_NET) and also update the trip stops. If the trip screen does not get a response within a certain amount of time (?GEO_DT_PCMS_TIMEOUT?) then it updates the request to be timed out (TIMEDOUT = ?Y?) and then tries the straight line calculation (Pythag_Calc) for updating its trip stops.

Code also already exists within MTS which takes the updated requests (REQ_STATUS = ?PROCESSED?) and then updates the relevant information within MTS. It moves the requests onto REQ_STATUS = ?APPLIED?.

This code is again going to remain unchanged and will continue to work as now.

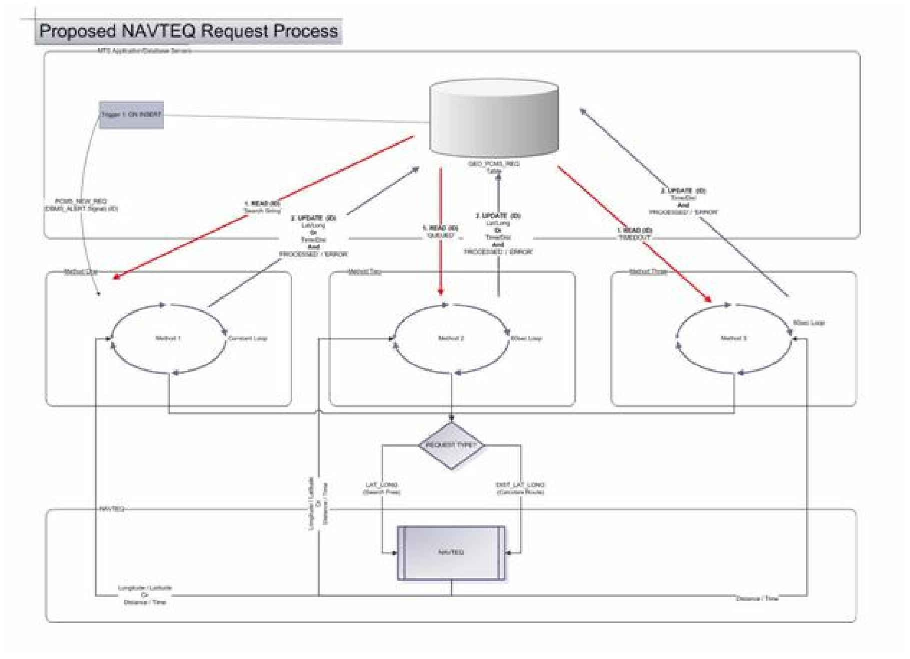
NB) The Location data is updated by the same trigger (T_GEO_PCMS_REQ_PROC_SIGNAL) as detailed above and network table is updated by the trip screens if the requests are processed in time.

A database job also exists which goes through the ?PROCESSED? requests for ?DIST_LATLONG? that were timed out (TIMEDOUT= ?Y?) on the trip screens and then updates the network table so that future requests can be satisfied directly from the table without need to call an external source.

This again can remain unchanged.

5.1 New Process for PCMS Requests

The new process flow will replace the VB and Map Point processes from Map Point process flow diagram (area within dotted lines) with the new NAVTEQ processes as follows :-


Proposed NAVTEQ Request Process
MTS Application/Database Servers
Trigger 1: ON INSERT
GEO_PCMS_REQ Table
PCMS_NEW_REQ (DBMS_ALERT Signal) (ID)
1. READ (ID) Search String
2. UPDATE (ID) Lat/Long Or Time/Dist And PROCESSED / ERROR
1. READ (ID) 'QUEUED'
2. UPDATE (ID) Lat/Long Or Time/Dist And PROCESSED / ERROR
1. READ (ID) 'TIMEDOUT'
2. UPDATE (ID) Time/Dist And PROCESSED / ERROR
Method One
Method Two
Method Three
Method 1
Constant Loop
Method 2
60sec Loop
Method 3
60sec Loop
REQUEST TYPE?
LAT_LONG (Search Free)
DIST_LAT_LONG (Calculate Route)
Longitude / Latitude Or Distance / Time
NAVTEQ
NAVTEQ
Longitude / Latitude Or Distance / Time
Distance / Time

The new code to process the PCMS requests using the NAVTEQ website will be triggered in one of 3 ways.

There are several scenarios that this needs to be covered :-

A request for information is made (either for a location or a trip), the main process is waiting for this signal and the process obtains the required information from NAVTEQ within a suitable period of time. This is dealt with by **Method One** in the process flow diagram above and is described in more detail below as The **First Method**.

A request for information is made (either for a location or a trip) and the main process is busy processing a previous request so the signal is ignored OR The main process receives the signal but fails to get a response from NAVTEQ within a sensible period of time. These are dealt with by **Method Two** in the process flow diagram and it is described in more detail below as The **Second Method**.

A request for information is made for a trip (after it has already checked the Network table and the reverse network entry). This is then timed out whilst waiting for the response to this request so it then uses the straight line calculation in order for the client to proceed with trip planning. This is dealt with by **Method Three** in the process flow diagram and is described in more detail below as The **Third Method**.

5.1.1 First Method

The ideal one and means that the request has been processed successfully within a very short period of time.

Whenever a request is added to the GEO_PCMS_REQ table (after insert trigger on the table - T_GEO_PCMS_REQ_ID) then a dbms_alert.signal called ?PCMS_NEW_REQ? is generated with the request ID as the message.

A permanently looping database job will exist that is waiting for the next signal. Once the dbms_alert is received then it will use the ID in the message to retrieve the appropriate request.

If this request is still at status ?QUEUED? then it will call the appropriate process for ths request ID (see later) to obtain the required information from NAVTEQ depending on the type of request.

If the process is still dealing with a request when the next dbms_alert.signal for a new request is sent then this signal will be never be received as the process is not a point in its code where it is waiting for the dbms_alert.

Also if the above process fails to get a response back from NAVTEQ within a sensible amount of time (depending upon the http transfer timeout setting) then the request will have been left at status ?QUEUED?.

In either of the above 2 cases when request is still at status ?QUEUED? then the second method will come into effect.

5.1.2 Second Method

Another database job that runs at regular intervals which loops through all of the requests that are still at status ?QUEUED? and older than a certain amount of time (to avoid picking up the one currently being processed by the first method) and also has not been TIMEDOUT (dealt with under method 3).

It will then call the same appropriate NAVTEQ process as the first method for each of the required request ID?s.

When a request from one of the trip screens is sent then the form waits for a pre-defined amount of time (?GEO_DT_PCMS_TIMEOUT?) for a DBMS_ALTER.Signal (type PCMS_REQ_PROCSSED_<request ID>) to be received back.

NB) This signal is generated by a trigger (T_GEO_PCMS_REQ_PROC_SIGNAL) on the request table whenever it is changed to status ?PROCESSED?

If a signal is received within this time then it will update the network table (GEO_NET) along with the trip stops but if it does not then it will update the TIMEDOUT field to ?Y? on the request.

5.1.3 Third Method

QUEUED? for type ?DIST_LATLONG? only and were TIMEDOUT = ?Y?.

It will then call the appropriate NAVTEQ process for this request ID.

The second and third method may be able to be combined into a single database job but we need to ensure that we do not interfere with the main process as it is the Trip screen that sets the TIMEDOUT flag not the first method.

The 2 types of requests (LATLONG and DIST_LATLONG) generated.

Each of the request types will need is dealt with differently by NAVTEQ so will each have their own new process.

The LATLONG request will generate a SearchFree request for NAVTEQ and extract the Longitude and Latitude values from the response.

The DIST_LATLONG request will generate a CalculateRoute request for NAVTEQ and extract the distance and time from the response.

These are dealt with in more detail in the following sections

5.2 New Lat/Long Request

When a new location is entered in MTS or its address is amended then a request on the GEO_PCMS_REQ table is generated (REQUEST_TYPE = ?LATLONG?) to obtain its Longitude and Latitude values from an external source (currently Map Point, previously PCMiler and being changed to be NAVTEQ).

The new process will receive the request (REQUEST_ID) that needs processing.

It will firstly confirm that the request is of the correct type (REQUEST_TYPE = ?LATLONG?) and that it is still waiting to be processed (REQ_STATUS = ?QUEUED?).

It will then build a Search_Free HTML request within a SOAP envelope using the information stored in the FROM_REF field for sending to NAVTEQ.

The {Login} value will be replaced by the Map24 login provided by NAVTEQ. This should be stored in a system registry (?GEO_NAVTEQ_LOGIN?).

The {Postcode} will be the extracted from search string provided in the FROM_REF field on the request record.

The utility UTL.GET_FIELD_VALUE will be used to retrieve the 5th field (always post code) out of the FROM_REF and pass it on to NAVTEQ in the search string.

This request will then be submitted to the NAVTEQ website using the http commands built into oracle and it will wait for a response.

NB) We may need to alter the set transfer timeout for http (using utl_http.set_transfer_timeoutfrom its current 60 seconds to maybe 5 seconds or lower to avoid tying up the process for too long on a single request and to keep it in line with the timeout from the Trip screen before it switches to the straight line calculation.

NB) May also need to set the proxy server for http (using utl_http.set_proxy).

If a successful response is obtained then the Longitude and Latitude fields will be extracted from the response.

The extracted Longitude and Latitude values will not be in exactly the correct format for storing on the request so will need some manipulation (divide by 60 as they will be returned in minutes rather than degrees and also round to 5 decimal places) to keep them in exactly the same format as now.

These will then be used to update the corresponding LATITUDE and LONGITUDE fields on the appropriate request along with updating the REQ_STATUS to ?PROCESSED?.

If a timeout is obtained then the REQ_STATUS will be left at ?QUEUED? so that the request can be dealt with again later.

If the response obtained does not contain the Longitude and Latitude tags then it will be assumed to be an error and the status will be set to ?ERROR? and the LAST_ERROR field will be set to an appropriate value.

5.3 New DIST_LATLONG Request

When a new trip stop is generated then the distance between the previous stop and the new stop and also the time to reach the new stop from the previous stop are required by the trip screens. This generates a request on the GEO_PCMS_REQUEST table (REQUEST_TYPE = ?DIST_LATLONG?) for information from an external source.

The new process will receive the request (REQUEST_ID) that needs processing.

It will firstly confirm that the request is of the correct type (REQUEST_TYPE = ?DIST_LATLONG?) and that it is still waiting to be processed (REQ_STATUS = ?QUEUED?).

It will then build a CalculateRoute HTML request within a SOAP envelope using the information stored in the FROM_REF and TO_REF fields for sending to NAVTEQ.

{login information} is the same as the SearchFree request (GEO_NAVTEQ_LOGIN).

FROM_REF and TO_REF will be provided in the format ?0532746N,0024205W?. Latitude is positive when N(orth) and negative when S(outh). Longitude is positive when W(est) and negative when E(ast). The format provided is DDDMMSSX. i.e. 3 digits for degrees, 2 digits for minutes, 2 digits for seconds and a character for N/S/E/W.

These will need manipulating in order to build the 4 parameters required by NAVTEQ called {long from}, {lat from}, {long to} and {lat to} in the above code.

They are all required in the same format as the data provided in the LATLONG request. Positive or negative minutes to lots of decimal places (upto 14 ?).

i.e. Convert using (DDD * 60) + MM + (SS / 60) and then sign based upon X.

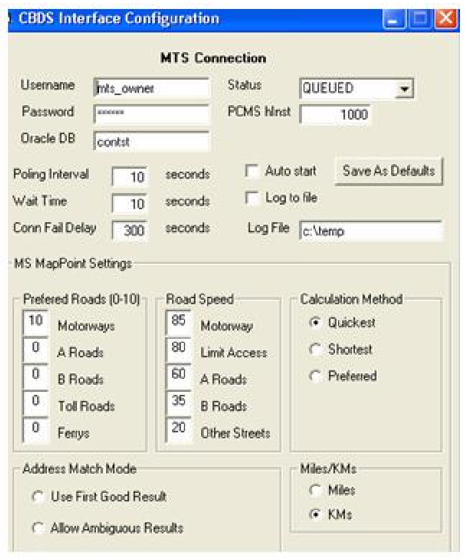
Also included in the request is the SpeedClassification definition.

The defaults within NAVTEQ are :- Motorway = 120, CountryRoad = 60 and CityRoad = 40.

In our request we have had coded (80, 60 and 50) which will become MTS defaults.

These can also optionally be setup as system registry values (GEO_NAVTEQ_MOTOR_SPEED, GEO_NAVTEQ_COUNTRY_SPEED and GEO_NAVTEQ_CITY_SPEED) to allow them to be maintained by the user instead of our default settings

Additional options currently available within Map Point are ;-



CBDS Interface Configuration

MTS Connection

Username: mts_owner
Password: ******
Oracle DB: contst
Status: QUEUED
PCMS hInst: 1000
Poling Interval: 10 seconds
Wait Time: 10 seconds
Conn Fail Delay: 300 seconds
[ ] Auto start
[ ] Log to file
Log File: c:\temp

MS MapPoint Settings

Prefered Roads (0-10): Motorways 10, A Roads 0, B Roads 0, Toll Roads 0, Ferrys 0

Road Speed: Motorway 85, Limit Access 80, A Roads 60, B Roads 35, Other Streets 20

Calculation Method: [✓ Quickest] Shortest Prefered

Address Match Mode: Use First Good Result / Allow Ambiguous Results

Miles/KMs: Miles [✓ KMs]

There is no NAVTEQ equivalent of preferred roads but they do have a calculation method (CalculationMode of Fastest or Shortest) which could also be added as a registry option (GEO_NAVTEQ_FAST_SHORT - ?Fastest?,?Shortest?) defaulting to ?Fastest? if not setup. A further option unique to NAVTEQ is the VehicleType which defaults to ?Car? but an option of ?Truck? exists which should be more applicable to DHL (it is assumed that it avoids narrow roads and low bridges but this needs further clarification from NAVTEQ).

There is no NAVTEQ equivalent of first or best result for addresses and rather than allowing NAVTEQ do a kms/miles calculation MTS will always use kms and then use the already existing registry setting ?GEO_DT_DISTANCE_UNITS' to determine how the values returned should be converted (see later).

The request will then be sent to NAVTEQ using the same commands as the previous request.

If successful then the response will contain then the distance (TotalLength) and time (TotalTime).

The TotalLength tag needs extracting (in metres) and converting before storing in TOTAL_KM on the request in either kilometres or miles depending on the system parameter ?GEO_DT_DISTANCE_UNITS'.

For conversion to kilometres simply dividing by 1,000 (will give 3 decimal places) and for miles divide by 1,609.3 and then round to 3 decimal places to keep data consistent with the current code.

The TotalTime tag need extracting (in seconds) and converting before storing in TOTAL_MINS on the request.

For conversion simply divide by 60 and round to the nearest minute.

The REQ_STATUS also needs updating on the request to ?PROCESSED?.

If a timeout is obtained then the REQ_STATUS will be left at ?QUEUED? so that the request can be dealt with again later.

If the response obtained does not contain the TotalLength and TotalTime tags then it will be assumed to be an error and the status will be set to ?ERROR? and the LAST_ERROR field will be set to an appropriate value.

6 References

Not Available

7 Document History

Version	Date	Status	Reason	Initials
1a	08/01/09	Draft	Initial version	DRM
1b	15/01/09	Draft	Initial Review	MJC
1c	16/01/09	Draft	Updated Draft	DRM
1d	19/01/09	Draft	Reviewed	MJC
1e	20/01/09	Draft	Updated Draft	DRM

8 Authorised By

Matt Crisford	Development Manager
Suk Sandhu	TMSCC MTS Product Manager