266060

Contents

1 266060

2 266060 - NW-7SDUK7/ New MTS to Paragon Interface

3 FUNCTIONAL OVERVIEW

3.1 Client Requirement

New interface from MTS to Paragon.

A revised interface format between MTS and Paragon is required as part of the M&S re-deployment, this should be based on the current MTS XML format and the Paragon format. The interface should be via ESI, assuming a server based version of Paragon, and be able to be managed via a new form in MTS to manage the data output to Paragon and then a new Interface Errors form to manage the data input from Paragon.

Once sent to Paragon orders should take the status of New and should not be modifiable, when received back from Paragon the status should be updated to either Unscheduled, Sched-Coll or Scheduled dependent on what Paragon has done with them. Only orders in the aforementioned statuses should be available to be exported to Paragon in the first place.

3.2 Solution

The solution will be based on MTS sending a schedule of orders to Paragon and Paragon responding with route, trip and stop information.

MTS outbound message to Paragon - Orders will be sent from MTS on demand using an upgraded version of the Paragon Interface form. This form allows orders to be selected using schedule, cost centre, customer and group name. As orders are included into the interface to paragon, they will be locked in MTS with status NEW until paragon responds. The Paragon interface form will show an audit trail of each interface export file created at summary and detail.

The file creation logic will reference a table of master data defining the expected ?journey legs? between collection and delivery locations. This master data will be maintained in MTS using a modified version of the Paragon X-Dock paths form and will be called Paragon Staging Posts. For each from and to location, a series of one or more legs can be defined in sequence. For each leg, the master data will hold the staging location code (the next destination), the distance and drive time, the minimum turnaround time and a (optional) default trailer type.

As the interface runs it will consider all orders for the output file for the requested schedule and where the order status is UNSCHEDULED or SCHED_COLL. It will also consider orders belonging to any schedule of the preceding 7 days (this will be a system parameter days setting) where a journey leg of the order is unplanned and calculated to start in the requested schedule for the interface.

The objective is to build the interface file to Paragon with all order journey legs that start in the selected schedule. The interface schedule will usually be today as the interface is likely to be run early hours of the morning for today?s trips. There needs to be a planning horizon cut-off. This will be controlled by a system parameter that allows a value of hours forward from the run time of the interface file.

Note that this assumes transport can run 24/7 and that Paragon manage non-working days or transport down-time.

If no Staging Post data is found, the order will be assumed to route directly as a single journey leg.

The interface file will be created in an XML format according to the MTS TripOrder.xsd.

MTS inbound message from Paragon - Paragon will optimise and schedule orders into routes, trips and stops. Every order (order leg) that is planned in Paragon will be interfaced back to MTS. It is understood that functionality in Paragon will allow routes to be ?cherry picked? to be scheduled. This means orders not planned by Paragon must automatically be reverted to their original status from NEW. This gives control of the order back into MTS.

ESI will transform the Paragon message into the MTS TripOrder.xsd XML format. A route in Paragon will become a trip in MTS. The format of the XML messages allows many trips to be included in one file and for each trip the sequence of trip stops and order activity at each stop to be defined. It is important that in the MTS XML file an order is shown 2 times on each trip, once for the load activity and once for the unload activity - this is regardless of the respective location types.

MTS will process the XML trip plan message. Using the information it will automatically create trips at planned status and allocate the appropriate orders to those trips. All the planned dates and times of each stop will be generated from the Paragon response including any defined break times.

It will be possible that for a multi-leg routing, Paragon might for one interface plan some but not all legs of an order. Ideally the legs should be planned in sequence by Paragon. MTS will recognise where not all legs are planned and set the order status to SCHED_COLL awaiting the next interface exchange or manual intervention.

The Business requires that a trip in MTS should represent a driver shift. This means aligning Paragon routes with MTS trips. One implication of this is that a trip in MTS might finally represent multiple transport operations where for example, the driver returns to base for a second load, or is requested to facilitate a collection / return / backload as additional instructions. As mentioned above, it is expected that the Paragon interface can be run more than once per day. This means that Paragon could optimise work as an extension of an existing route. MTS must therefore have the capability of updating an existing trip with additional stops from a subsequent interface exchange with Paragon. In simple terms, for own fleet this might mean adding extra stops and rescheduling the return to base (CL) until later in the drivers day. MTS will need to validate that changes to trips don?t conflict with execution updates indicating completed operations from Isotrak.

MTS will provide an interface errors form to allow Paragon messages to be reviewed and any validation errors / failures identified.

The trip sum and trip manipulation forms will be changed to enhance the FIND function. The Paragon route number will be stored in the route field on MTS. The FIND function will allow this route number (within a schedule) to be found and all the trip details displayed.

3.3 Scope

This change will be applied to system version 10.6.

4 FUNCTIONAL DESCRIPTION

4.1 Paragon Staging Posts

A new table will be created to hold the new paragon staging post data called

PAR_STAGING_POSTS :-

FROM_LOC VARCHAR2(12)

TO_LOC VARCHAR2(12)

XDOCK_SEQ NUMBER(2)

XDOCK_LOC VARCHAR2(12)

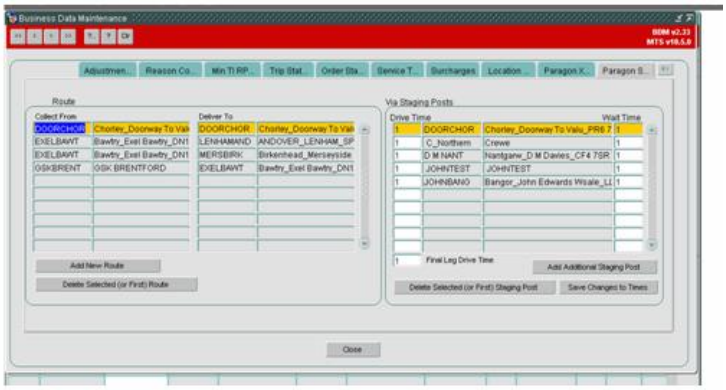
DRIVE_TIME NUMBER(5)

WAIT_TIME NUMBER(5)

Unique primary key contains FROM_LOC, TO_LOC and XDOCK_SEQ

4.1.1 Business Data Form

A new tab will be added to the Business Data Maintenance (BDM) form to maintain this new data in a similar way to the existing Paragon Cross-Dock tab :-



The form will be split into 2 sections :-

The header information will contain :-

FROM_LOC : Original loading location for the order

TO_LOC : Final delivery location for the order

And the detail information :-

XDOCK_LOC : The location ID for the cross-dock (Staging post)

DRIVE_TIME : The minimum driving time from the previous location to this location

WAIT_TIME : The minimum turn around time at the cross-dock location

NB) The times provided will be in minutes and must be sensible values.

The XDOCK_SEQ will be populated behind the scenes and is simply a row count which will need to be updated whenever cross-dock details are added or deleted.

The last leg of the journey will always be XDOCK_SEQ = 99 and will not need a XDOCK_LOC or a WAIT_TIME.

i.e. From the last cross-dock location to the final delivery location.

4.1.2 New CSV Import

file produced by the client and upload it into the new file.

It will use the standard Import forms (IMPORTS_MAINT and IMPORTS_EXEC) so the necessary data for the new import will nee to be added to the

The new code will be added to the package IMP and will be called PAR_NEW_XDOCKS.

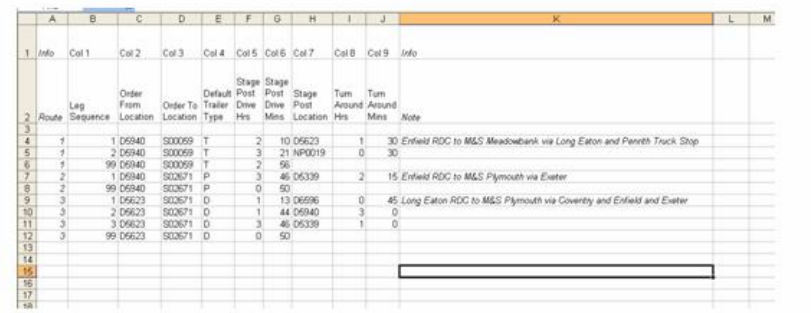
The csv will need to create PAR_STAGING_POSTS records if one does not already exist otherwise it needs to update it (possibly easier to delete them all and re-create them in a similar fashion to slots import).

The 3 locations will all need to be validated as valid locations in MTS and the trailer type also validated against the data in MTS.The drive and wait times must also be sensible values in minutes.

The XDOCK_SEQ may be provided (ensure it is unique within FROM_LOC and TO_LOC) as should the XDOCK_SEQ but if not provided then count manually.

NB) A record must be provided without a XDOCK_LOC for each combination of FROM_LOC and TO_LOC and this must have sequence 99 or be assigned it.

A sample CSV may look like :-

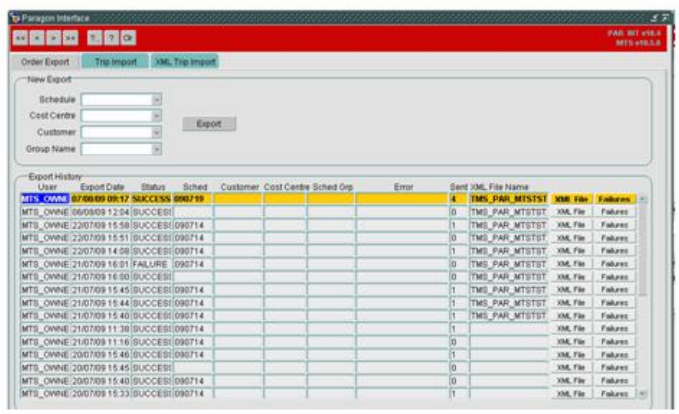


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Info	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Info		
2	Route	Leg Sequence	Order From Location	Order To Location	Default Trailer Type	Stage Post Drive Hrs	Stage Post Drive Mins	Stage Post Location	Turn Around Hrs	Turn Around Mins	Note		
3													
4	1	1	D6940	S00059	T	2	10	D5623	1	30	Enfield RDC to M&S Meadowbank via Long Eaton and Penrith Truck Stop		
5	1	2	D6940	S00059	T	3	21	NP0019	0	30			
6	1	99	D6940	S00059	T	2	56						
7	2	1	D6940	S02671	P	3	46	D5339	2	15	Enfield RDC to M&S Plymouth via Exeter		
8	2	99	D6940	S02671	P	0	50						
9	3	1	D5623	S02671	D	1	13	D6596	0	45	Long Eaton RDC to M&S Plymouth via Coventry and Enfield and Exeter		
10	3	2	D5623	S02671	D	1	44	D6940	3	0			
11	3	3	D5623	S02671	D	3	46	D5339	1	0			
12	3	99	D5623	S02671	D	0	50						
13													
14													
15													
16													
17													

4.2 XML Outbound

4.2.1 Paragon Interface form Outbound Orders

The paragon kick-off screen (PAR_INT) for the Orders Export will look like :-



The form itself will not change much for Order Export as the same parameters are being used and the same audit records

will be created.

An additional field called file name will display the name of the file that was produced and sent to Paragon containing the XML for the orders.

Also an additional 2 buttons will be available for each record on the cluster :-

?XML File? to show the XML file produced by the package that should still be in the archive area (NB/ Very old exports will have been archived).

?Failures? to show the failures file produced by the package that should be in the failures folder (name has ?_FAIL? at the end of the file name and is a TXT rather than XML file) and contain a list of orders that passed the initial selection criteria but that could not be sent to Paragon as valid time windows could not be calculated for each outstanding journey leg.

The only other change will be behind the Export button.

The existing system registry called ?PARAGON_INSTALLED? which is currently set to either ?YES? or ?NO? will be extended so that it can now also be set to ?XML? for when the new XML version of paragon is installed.

If the system registry is ?YES? then the existing code will be called unchanged (CSV.Orders_to_Paragon) but if it is set to ?XML? then the new code will be called instead (PAR.XML_Orders_to_Paragon).

4.2.2 PAR.XML_Orders to Paragon

The new procedure will receive the 4 values set in the form.

It will also need to check the new system registry ?PAR_XML_OLD_ORDERS? to determine how many days in the past old orders are also going to be checked and possibly re-sent to Paragon.

The filename will be in the format ?TMS_PAR_?<database>?_?<date/time>.XML? and directory path is determined using the system registry ?PAR_OUTBOUND_PATH?. A safe copy if the file will be stored in ?PAR_OUTBOUND_ARCH?. It will then be automatically FTP?d using the registries :- PAR_FTP_DESTINATION_IP_ADDRESS,

PAR_FTP_DESTINATION_PORT,

PAR_FTP_DESTINATION_DIRECTORY,

PAR_FTP_DESTINATION_USERNAME,

PAR_FTP_DESTINATION_PASSWORD and then deleted from the outbound folder.

As the interface runs it will consider all orders for the output file for the requested schedule where the order status is UNSCHEDULED or SCHED_COLL. It will also consider UNSCHEDULED or SCHED_COLL orders belonging to any schedule of the preceding number of days determined using the above parameter.

?? Journey Legs calculated to start in the requested schedule for the interface.

The file creation logic will reference the new tables (PAR_STAGING_POSTS_HDR and PAR_STAGING_POSTS_DTL) to obtain the journey leg data (using FROM_LOC and TO_LOC from the order) defining the expected ?journey legs? between collection and delivery locations (including the header record for the final leg drive time and getting the details in XDOCK_SEQ order for intermediate locations, drive and wait times).

The objective is to build the interface file to Paragon with all outstanding order journey legs that can be done on the selected schedule. The interface schedule will usually be today as the interface is likely to be run early hours of the morning for today?s trips.

?? There needs to be a planning horizon cut-off. This will be controlled by a system parameter that allows a value of hours forward from the run time of the interface file.

The UNSCHEDULED orders will need to take all journey legs into account but the SCHED_COLL orders will need to check the CURRENT_DEPOT field to find the orders current location and only take into account the journey legs after this location.

The data retrieved will be used to calculate appropriate time windows for the outstanding journey legs and to populate the standard XML format according to the MTS TripOrder.xsd.

NB) each journey leg will represent a new order as far as the paragon interface is concerned.

(See appendix A for an example layout and appendix B for the full definition).

The <TMS_REF> value will be the presented as the MTS OMS_REF (the unique internal order number) + (- sign as delimiter) + the leg numerical sequence number (XDOCK_SEQ).

This will ensure a unique order reference number is available to map into the Paragon format by ESI.

The Paragon interface file requires earliest and latest despatch and earliest and latest arrive for each journey leg. MTS will calculate these time windows automatically leg by leg using the order window date times, the leg drive times and minimum turnaround time at the X-Dock locations. (If previous legs have been planned and perhaps actual date times of departure and arrival are known, these date times will be used to calculate for the next leg).

NB) All of the other XML fields are self explanatory and will use the same code as the existing order outbound message to LOTS.

As well as writing the data to the XML file (using system registries) it will also need to write a record to the PAR_EXPORT_HISTORY table for display on the PAR_INT form.

See examples below of all possible scenarios that we will code for when determining number of orders (journey legs) and the valid windows for each leg.

Examples :-

1. Direct (A-B)
2. Single Cross-Dock for Unscheduled Order (A-B-C)
3. Double Cross-Dock for Unscheduled Order (A-B-C-D)
4. Triple (or 4 or 5) Cross-Docks for Unscheduled Order (A-B-C-D-E)
5. Single Cross-Dock for SCHED_COLL Order (B-C)
6. Double Cross-Dock for SCHED_COLL Order at first cross-dock (B-C-D).
7. Double Cross-Dock for SCHED_COLL Order at second cross-dock (C-D).
8. Triple Cross-Dock for SCHED_COLL Order at second cross-dock (C-D-E).
9. Triple (or more) Cross-Docks for SCHED_COLL Order at first cross-dock (B-C-D-E).

4.2.2.1 Examples of a Direct

No journey legs will be defined on PAR_STAGING_POSTS for FROM_LOC, TO_LOC.

Order to be collected between 09:00 and 10:00 on 29/06/09 and delivered between 14:00 and 16:00 on the same day.

The XML will contain the same values as the original order :-

<TMS_REF>OMSREF</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-09T09:00:00</EARLY_AVAIL_DATE>

<LATE_AVAIL_DATE>2009-06-09T10:00:00</LATE_AVAIL_DATE>

<EARLY_DEL_DATE>2009-06-09T14:00:00</EARLY_DEL_DATE>

<LATE_DEL_DATE>2009-06-09T16:00:00</LATE_DEL_DATE>

4.2.2.2 Example of Single Cross-Dock for Unscheduled Order

Collect A, Cross-Dock to B and final delivery to C.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 3 Hours (A-B)

Wait Time at B is 1 hour

Seq 99, Drive Time 2 Hours (B-C)

Order to be collected between 06:00 and 07:00 on 29/06/09 and delivered between 19:00 and 20:00 on the same day.

The first calculation to perform is the same one that needs to be performed for all of the examples except Directs or SCHED_COLL with only one journey leg outstanding and that is to confirm that the windows provided do not make the trip impossible.

Take Late Depart A (07:00) add all drive and wait times (3 + 1 + 2) to give 13:00 and this must be before Early Arrive C (19:00) otherwise give an error stating that the windows need adjusting as the trip is not possible as it has overlapping windows.

Calc 1 is Early Depart A (06:00) + Drive Time A-B (3 Hours) = 09:00 Early Arrive B

Calc 4 is Late Arrive C (20:00) - Drive Time B-C (2 Hours) = 18:00 Late Depart B

Calc Late Depart A (07:00) + Drive Time A-B (3 Hours) = 10:00 Late Arrive B

Calc Early Arrive C (19:00) - Drive Time B-C (2 Hours) = 17:00 Early Depart B

Difference is 7 Hours - Wait at B (1 Hour) = 6 Hours slack

Calc 2 is Late Arrive B (10:00) + Slack (3 Hours) = 13:00 New Late Arrive B

Calc 3 is Late Depart B (17:00) - Slack (3 Hours) = 14:00 New Early Depart B

NB) Slack time is split evenly between journey legs so can arrive at B later and depart earlier.

The XML will contain 2 orders :-

A-B Journey Leg

<TMS_REF>OMSREF-1</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T06:00:00</EARLY_AVAIL_DATE> As Order

<LATE_AVAIL_DATE>2009-06-29T07:00:00</LATE_AVAIL_DATE> As Order

<EARLY_DEL_DATE>2009-06-29T09:00:00</EARLY_DEL_DATE> Calc 1

<LATE_DEL_DATE>2009-06-29T:13:00</LATE_DEL_DATE> Calc 2

AND

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T14:00:00</EARLY_AVAIL_DATE> Calc 3

<LATE_AVAIL_DATE>2009-06-29T18:00:00</LATE_AVAIL_DATE> Calc 4

<EARLY_DEL_DATE>2009-06-29T19:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-29T20:00:00</LATE_DEL_DATE> As Order

4.2.2.3 Example of Double Cross-Dock for Unscheduled Order

Collect A, Cross-dock via B and C with final delivery to D.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 5 Hours (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 4 Hours (B-C)

Wait Time at B is 2 hours

Seq 99, Drive Time 6 Hours (C-D)

Order to be collected between 09:00 and 10:15 on 29/06/09 and delivered between 12:00 and 13:00 on the following day.

Take Late Depart A (29/06 10:15) add all drive and wait times (5 + 1 + 4 + 2 + 6) to give 30/06 04:15 (the following day) and this must be before Early Arrive D (30/06 12:00) otherwise give an error stating that the windows need adjusting as the trip is not possible as it has overlapping windows.

Forwards from A :-

Calc 1 : Early Depart A (29/06 09:00) + Drive Time A-B (5 Hours) = 14:00 Early Arrive B

Calc 2 : Late Depart A (29/06 10:15) + Drive Time A-B (5 Hours) = 15:15 Late Arrive B

Calc 3 : Late Arrive B (29/06 15:15) + Wait B (1 Hour) = 16:15 Early (Dep B + Arrive C)

Backwards from D :-

Calc 6 : Late Arrive D (30/06 13:00) - Drive Time C-D (6 Hours) = 07:00 Late Depart C

Calc 5 : Early Arrive D (30/06 12:00) - Drive Time C-D (6 Hours) = 06:00 Early Depart C

Calc 4 : Early Depart C (30/06 06:00) - Wait C (2 Hours) = 04:00 Late (Dep B + Arr C)

NB) All of slack time will be given to middle leg.

The XML will contain 3 orders :-

A-B Journey Leg

<TMS_REF>OMSREF-1</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T09:00:00</EARLY_AVAIL_DATE> As Order

<LATE_AVAIL_DATE>2009-06-29T10:15:00</LATE_AVAIL_DATE> As Order

<EARLY_DEL_DATE>2009-06-29T14:00:00</EARLY_DEL_DATE> Calc 1

<LATE_DEL_DATE>2009-06-29T:15:15</LATE_DEL_DATE> Calc 2

AND

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T16:15:00</EARLY_AVAIL_DATE> Calc 3

<LATE_AVAIL_DATE>2009-06-30T04:00:00</LATE_AVAIL_DATE> Calc 4

<EARLY_DEL_DATE>2009-06-29T16:15:00</EARLY_DEL_DATE> Calc 3

<LATE_DEL_DATE>2009-06-30T04:00:00</LATE_DEL_DATE> Calc 4

AND

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-30T06:00:00</EARLY_AVAIL_DATE> Calc 5

<LATE_AVAIL_DATE>2009-06-30T07:00:00</LATE_AVAIL_DATE> Calc 6

<EARLY_DEL_DATE>2009-06-30T12:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T13:00:00</LATE_DEL_DATE> As Order

4.2.2.4 Example of Triple (or more) Cross-Docks for Unscheduled Order

Collect A, Cross-dock via B,C and D with final delivery to E.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 1 Hour (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 3 Hours (B-C)

Wait Time at C is 1 hour

Seq 3, Cross Dock Location C

Drive Time 2 Hours (C-D)

Wait Time at D is 2 hours

Seq 99, Drive Time 6 Hours (D-E)

Order to be collected between 09:00 and 12:00 on 29/06/09 and delivered between 11:00 and 12:00 on the following day.

Take Late Depart A (29/06 12:00) add all drive and wait times (1 + 1 + 3 + 1 + 2 + 2 + 6) to give 30/06 04:00 (the following day) and this must be before Early Arrive D (30/06 11:00) otherwise give an error stating that the windows need adjusting as the trip is not possible as it has overlapping windows.

Forwards from A :-

Calc 1 : Early Depart A (29/06 09:00) + Drive Time A-B (1 Hour) = 10:00 Early Arrive B

Calc 2 : Late Depart A (29/06 12:00) + Drive Time A-B (1 Hour) = 13:00 Late Arrive B

Calc 3 : Late Arrive B (29/06 13:00) + Wait B (1 Hour) = 14:00 Early (Dep B + Arrive C)

Backwards from E :-

Calc 8 : Late Arrive E (30/06 12:00) - Drive Time D-E (6 Hours) = 06:00 Late Depart D

Calc 7 : Early Arrive E (30/06 11:00) - Drive Time D-E (6 Hours) = 05:00 Early Depart D

Calc 6 : Early Depart D (30/06 05:00) - Wait D (2 Hours) = 03:00 Late (Dep C + Arr D)

Difference between Early Arrive C (26/06 14:00) and Late Depart C (27/06 03:00) is 13 Hours less wait at C (1 hour) gives 12 hours slack time.

Middle leg drive times from B-C (3 Hours) + C-D (2 Hours) = 5 hours

Assign slack so each hour drive becomes 12 / 5 = 2.4 Hours (2 Hours 24 minutes).

Calc 4 : Early Dep B (29/06 14:00) + Drive B-C (3 * 2.4) = 21:12 Late (Dep B + Arr C)

Calc 5 : Late Arrive C (21:12) + Wait C (1 Hour) = 22:12 Early (Dep C + Arr D)

NB) Slack time is proportionally assigned to the middle legs based upon the drive times.

The XML will contain 4 orders :-

A-B Journey Leg

<TMS_REF>OMSREF-1</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T09:00:00</EARLY_AVAIL_DATE> As Order

<LATE_AVAIL_DATE>2009-06-29T12:00:00</LATE_AVAIL_DATE> As Order

<EARLY_DEL_DATE>2009-06-29T10:00:00</EARLY_DEL_DATE> Calc 1

<LATE_DEL_DATE>2009-06-29T:13:00</LATE_DEL_DATE> Calc 2

AND

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T14:00:00</EARLY_AVAIL_DATE> Calc 3

<LATE_AVAIL_DATE>2009-06-29T21:12:00</LATE_AVAIL_DATE> Calc 4

<EARLY_DEL_DATE>2009-06-29T14:00:00</EARLY_DEL_DATE> Calc 3

<LATE_DEL_DATE>2009-06-29T21:12:00</LATE_DEL_DATE> Calc 4

AND

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T22:12:00</EARLY_AVAIL_DATE> Calc 5

<LATE_AVAIL_DATE>2009-06-30T03:00:00</LATE_AVAIL_DATE> Calc 6

<EARLY_DEL_DATE>2009-06-29T22:12:00</EARLY_DEL_DATE> Calc 5

<LATE_DEL_DATE>2009-06-30T03:00:00</LATE_DEL_DATE> Calc 6

AND

D-E Journey Leg

<TMS_REF>OMSREF-4</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-30T05:00:00</EARLY_AVAIL_DATE> Calc 7

<LATE_AVAIL_DATE>2009-06-30T06:00:00</LATE_AVAIL_DATE> Calc 8

<EARLY_DEL_DATE>2009-06-30T11:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T12:00:00</LATE_DEL_DATE> As Order

4.2.2.5 Example of Single Cross-Dock for Sched_Coll Order

Order has already been collected from A into the Cross-Dock location B so just needs the final delivery from B to C.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 3 Hours (A-B)

Wait Time at B is 1 hour

Seq 99, Drive Time 2 Hours (B-C)

Order already collected and transported to cross-dock B at 11:00 on 29/06/09 and to be delivered between 19:00 and 20:00 on the same day.

Check that the Actual Arrive B (11:00) + Wait B (1Hour) + Drive B-C (2 Hours) = 14:00 is earlier than Late Arrive C (20:00) otherwise give an error as trip is impossible.

NB) The Current_Depot on the order will be location B. Check the actual arrival time (or planned if actual is not set) on Sch_Trip_Stop for the order at location B.

Calc 1 is Actual Arrive at B (11:00) + Wait B (1 Hour) = 12:00 Early Depart B

Calc 2 is Late Arrive C (20:00) - Drive B-C (2 Hours) = 18:00 Late Depart B

NB) Slack time is all on departure time from B.

The XML will contain a single order :-

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T12:00:00</EARLY_AVAIL_DATE> Calc 1

<LATE_AVAIL_DATE>2009-06-29T18:00:00</LATE_AVAIL_DATE> Calc 2

<EARLY_DEL_DATE>2009-06-29T19:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-29T20:00:00</LATE_DEL_DATE> As Order

4.2.2.6 Example of Double Cross-Dock for SCHED_COLL with Order at first cross-dock (B-C-D)

Order has already been collected from A into the Cross-Dock location B so now needs the cross-dock to B-C and the final delivery to C-D.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 5 Hours (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 4 Hours (B-C)

Wait Time at C is 2 hours

Seq 99, Drive Time 6 Hours (C-D)

Order?s Current_Depot is B and it arrived there at 14:30.

Take Actual Arrive B (29/06 14:30) add all remaining wait and drive times (1 + 4 + 2 + 6) to give 30/06 03:30 must be before Early Arrive D (30/06 12:00) otherwise give an error stating trip is impossible.

Forwards from B :-

Calc 1 : Actual Arrive B (29/06 14:30) + Wait B (1 Hour) = 15:30 Early (Dep B + Arr C)

Backwards from D :-

Calc 4 : Late Arrive D (30/06 13:00) - Drive Time C-D (6 Hours) = 07:00 Late Depart C

Calc 3 : Early Arrive D (30/06 12:00) - Drive Time C-D (6 Hours) = 06:00 Early Depart C

Calc 2 : Early Depart C (30/06 06:00) - Wait C (2 Hours) = 04:00 Late (Dep B + Arr C)

NB) All of slack time will be given to B-C leg.

The XML will contain 2 orders :-

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T15:30:00</EARLY_AVAIL_DATE> Calc 1

<LATE_AVAIL_DATE>2009-06-30T04:00:00</LATE_AVAIL_DATE> Calc 2

<EARLY_DEL_DATE>2009-06-29T15:30:00</EARLY_DEL_DATE> Calc 1

<LATE_DEL_DATE>2009-06-30T04:00:00</LATE_DEL_DATE> Calc 2

AND

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-30T06:00:00</EARLY_AVAIL_DATE> Calc 3

<LATE_AVAIL_DATE>2009-06-30T07:00:00</LATE_AVAIL_DATE> Calc 4

<EARLY_DEL_DATE>2009-06-30T12:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T13:00:00</LATE_DEL_DATE> As Order

4.2.2.7 Example of Double Cross-Dock for SCHED_COLL Order at second cross-dock (C-D)

Order has already been collected from A into the Cross-Dock B, Trunked from Cross-Dock B to Cross-Dock C and now only needs the final delivery C-D.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 5 Hours (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 4 Hours (B-C)

Wait Time at C is 2 hours

Seq 99, Drive Time 6 Hours (C-D)

Order?s Current_Depot is C and it arrived there at 17:30 on 29/06.

Take Actual Arrive C (29/06 19:30) add all remaining wait and drive times (2 + 6) to give 30/06 01:30 must be before Early Arrive D (30/06 12:00) otherwise give an error stating trip is impossible.

Forwards from C :-

Calc 1 : Actual Arrive C (29/06 19:30) + Wait C (2 Hours) = 21:30 Early Depart C

Backwards from D :-

Calc 2 : Late Arrive D (30/06 13:00) - Drive Time C-D (6 Hours) = 07:00 Late Depart C

NB) All of slack time will be given to depart from C.

The XML will contain 1 order :-

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T21:30:00</EARLY_AVAIL_DATE> Calc 1

<LATE_AVAIL_DATE>2009-06-30T07:00:00</LATE_AVAIL_DATE> Calc 2

<EARLY_DEL_DATE>2009-06-30T12:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T13:00:00</LATE_DEL_DATE> As Order

4.2.2.8 Example of Triple Cross-Dock for SCHED_COLL Order at second cross-dock (C-D-E)

Order has already been collected from A into the Cross-Dock B, Trunked from Cross-Dock B to Cross-Dock C and now only needs the Trunk from cross-dock C to D and the final delivery D-E.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 1 Hour (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 3 Hours (B-C)

Wait Time at C is 1 hour

Seq 3, Cross Dock Location D

Drive Time 2 Hours (C-D)

Wait Time at D is 2 hours

Seq 99, Drive Time 6 Hours (D-E)

Order arrived at C (Current_Depot) on 29/06 at 16:00 (from corresponding Sch_Trip_Stop) and needs to be delivered between 11:00 and 12:00 on 30/06 (the following day).

Take Actual Arrive C (29/06 16:00) add all remaining wait and drive times (1 + 2 + 2 + 6) to give 30/06 03:00 must be before Early Arrive E (30/06 11:00) otherwise give an error stating trip is impossible.

Forwards from C :-

Calc 1 : Actual Arrive C (29/06 16:00) + Wait C (1 Hour) = 17:00 Early Depart C

Calc 2 : Early Depart C (17:00) + Drive C-D (2 Hours) = 19:00 Early Arrive D

Backwards from E :-

Calc 6 : Late Arrive E (30/06 12:00) - Drive Time D-E (6 Hours) = 06:00 Late Depart D

Calc 5 : Early Arrive E (30/06 11:00) - Drive Time D-E (6 Hours) = 05:00 Early Depart D

Calc 4 : Late Depart D (30/06 06:00) - Wait D (2 Hours) = 04:00 Late Arrive D

Calc 3 : Late Arrive D (04:00) - Drive C-D (2 hours) = 02:00 Late Depart C

NB) All slack assigned to C-D.

The XML will contain 2 orders :-

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T17:00:00</EARLY_AVAIL_DATE> Calc 1

<LATE_AVAIL_DATE>2009-06-30T02:00:00</LATE_AVAIL_DATE> Calc 3

<EARLY_DEL_DATE>2009-06-29T:19:00</EARLY_DEL_DATE> Calc 2

<LATE_DEL_DATE>2009-06-30T04:00:00</LATE_DEL_DATE> Calc 4

AND

D-E Journey Leg

<TMS_REF>OMSREF-4</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-30T05:00:00</EARLY_AVAIL_DATE> Calc 5

<LATE_AVAIL_DATE>2009-06-30T06:00:00</LATE_AVAIL_DATE> Calc 6

<EARLY_DEL_DATE>2009-06-30T11:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T12:00:00</LATE_DEL_DATE> As Order

4.2.2.9 Example of Triple Cross-Dock for SCHED_COLL Order at first cross-dock (B-C-D-E)

Order has already been collected from A into the Cross-Dock B and now needs the trunks from cross-dock B-C and C-D and the final delivery D-E.

PAR_STAGING_POSTS :- Seq 1, Cross Dock Location B

Drive Time 1 Hour (A-B)

Wait Time at B is 1 hour

Seq 2, Cross Dock Location C

Drive Time 2.5 Hours (B-C)

Wait Time at C is 0.5 hours

Seq 3, Cross Dock Location D

Drive Time 1 Hours (C-D)

Wait Time at D is 0.5 hours

Seq 99, Drive Time 6 Hours (D-E)

Order arrived at B (Current_Depot) on 29/06 at 11:12 (from corresponding Sch_Trip_Stop) and needs to be delivered between 11:00 and 12:00 on 30/06 (the following day).

Take Actual Arrive B (29/06 11:12) add all remaining wait and drive times (1 + 2.5 + 0.5 + 1 + 0.5 + 6) to give 29/06 22:42 must be before Early Arrive E (30/06 11:00) otherwise give an error stating trip is impossible.

Forwards from B :-

Calc 1 : Actual Arrive B (29/06 11:12) + Wait B (1 Hour) = 12:12 Early Depart B

Backwards from E :-

Calc 6 : Late Arrive E (30/06 12:00) - Drive Time D-E (6 Hours) = 06:00 Late Depart D

Calc 5 : Early Arrive E (30/06 11:00) - Drive Time D-E (6 Hours) = 05:00 Early Depart D

Calc 4 : Early Depart D (30/06 05:00) - Wait D (0.5 Hours) = 04:30 Late Arrive D

Difference between Early Depart B (29/06 12:12) and Late Arrive D (30/06 04:30) is 16 Hours 18 minutes less wait at C (30 minutes) gives 15 Hours 48 minutes as slack plus intermediate drive times (B-C and C-D).

Divide this by 3 Hours 30 minutes (actual drive times between B-C plus C-D) to give 4Hours and 31 minutes slack time equivalent for each drive hour (4.51 Hours).

Calc 2 : Early Dep B (29/06 12:12) + Drive B-C (2.5 * 4.51) = 23:29 Early Arr C

Calc 3 : Early Arr C (23:29) + Wait C (30 minutes) = 23:59 Early Dep C

NB) Early Dep C (23:59) + Drive C-D (1 * 4.51) = 04:30 (Same as Calc 4).

NB) Slack time is proportionally assigned to the middle legs based upon the drive times.

The XML will contain 3 orders :-

B-C Journey Leg

<TMS_REF>OMSREF-2</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T12:12:00</EARLY_AVAIL_DATE> Calc 1

<LATE_AVAIL_DATE>2009-06-29T:23:29:00</LATE_AVAIL_DATE> Calc 2

<EARLY_DEL_DATE>2009-06-29T:12:12:00</EARLY_DEL_DATE> Calc 1

<LATE_DEL_DATE>2009-06-29T23:29:00</LATE_DEL_DATE> Calc 2

AND

C-D Journey Leg

<TMS_REF>OMSREF-3</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-29T23:59:00</EARLY_AVAIL_DATE> Calc 3

<LATE_AVAIL_DATE>2009-06-30T04:30:00</LATE_AVAIL_DATE> Calc 4

<EARLY_DEL_DATE>2009-06-29T23:59:00</EARLY_DEL_DATE> Calc 3

<LATE_DEL_DATE>2009-06-30T04:30:00</LATE_DEL_DATE> Calc 4

AND

D-E Journey Leg

<TMS_REF>OMSREF-4</TMS_REF>

<EARLY_AVAIL_DATE>2009-06-30T05:00:00</EARLY_AVAIL_DATE> Calc 5

<LATE_AVAIL_DATE>2009-06-30T06:00:00</LATE_AVAIL_DATE> Calc 6

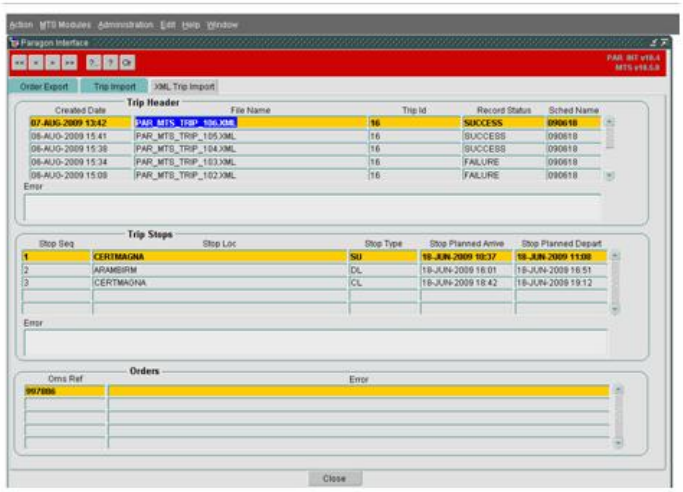
<EARLY_DEL_DATE>2009-06-30T11:00:00</EARLY_DEL_DATE> As Order

<LATE_DEL_DATE>2009-06-30T12:00:00</LATE_DEL_DATE> As Order

4.3 XML Inbound

4.3.1 Paragon Interface Form Inbound Trips

The paragon screen (PAR_INT) for the XML import of trips will look like :-



This is totally different from the existing options so a new tab will need writing and the configuration will be setup to allow either the old tab (csv manual import) or the new tab (never both).

There will be 3 levels :- Trip, Trip Stop and Order.

The top level will be a list of trips provided in the file and will be from the new PAR_XML_TRIP table. It will contain the status for the trip to indicate if it has been loaded successfully or not and if not then an appropriate error message will be shown.

The next level will be the trip stops and will be from the new table PAR_XML_TRIP_STOPS. It will also show any appropriate error messages generated at stop level.

The bottom level will be a list of orders that are either being loaded or unloaded at the stops and will be from the new table PAR_XML_TRIP_ORDER. It again will show any error messages generated at the order level.

4.3.2 PAR.INT_XML_Trips_From_Paragon

The current XML inbound process is split into 2 distinct stages:- The first updates the trip information and the second updates the order information.

The current paragon interface is also in 2 stages :- The first stage uses the orders to automatically generate the trips (resetting the status on the orders) and second updates the times on the generated trips.

Up until now all trip stops where generated by adding an order to the trip but for the new code this will no longer be possible as some stops may not even have orders (trailer pick ups).

The new code will need to :-

New database job running every 5 minutes selecting files of the correct name from the correct folder using system registries ?PAR_INBOUND_PATH? and ?PAR_INBOUND_IDENTIFIER? using the standard DP_FILE_HANDLING.GET_XML procedure.

The additional 2 parameters are obtained from system registries ?PAR_INBOUND_LISTING_NAME? and ?PAR_LISTING_SCRIPT_NAME?.

If the file is processed OK then it will be moved to the folder identified by system registry ?PAR_INBOUND_ARCH? otherwise it will be moved to the folder identified by the system registry ?PAR_INBOUND_FAIL?.

Extract the data out of the XML into the standard table XML_EXTRACT table using the standard package XML.XML_EXTRACT.

For every Trip (<TRIP_ID>) extracted then the code will need to do the following :-

Extract all of the trip header information (<TRIP_HEADER> and <TRIP_DETAIL> fields) and use them to write a record to the PAR_XML_TRIP table with appropriate validation for each field and for mandatory fields.

NB) <TRIP_SCHEDULE> will go into SCHED_NAME.

NB) If at any stage of the population of the intermediate tables an error occurs then it will need to update the error message on the appropriate level, roll this error back up to the trip level but continue loading the data into the intermediate data tables so that the PAR_INT screen has a full picture of what was in the XML.

For every trip stop (<STOP_SEQ>) within each trip then extract the trip stop information (<STOP_HEADER> and <STOP_DETAIL>) with appropriate validation for each field and for mandatory fields and put this data into the new table PAR_XML_TRIP_STOP.

For every OMS Ref (<TMS_REF>) within each trip stop then write a record into the new table PAR_XML_TRIP_ORD.

Once all of the interface data is written then another pass will then go through the recently created interface records to populate the real MTS data with any extra validation required.

For every PAR_XML_TRIP (PXT) for the file name currently being processed that does not already have an error message associated with it then :-

NB) If at any stage of this updating process an error occurs then it will need to update the error message on the appropriate level, roll this error back up to the trip level and then continue processing with the next trip.

Check if the data provided is for an existing Trip in MTS (link SCHED_NAME to SCHED_NAME and TRIP_ID to ROUTE_CODE on SCH_TRIP) or is a new trip.

NB) The Trip_ID provided by Paragon is simply a count that is reset at the start of each day (schedule). This will be stored in SCH_TRIP.ROUTE_CODE.

For new Trips then :-

Create a SCH_TRIP record using the standard code TRM.insert_trip passing the extracted schedule name, the hard coded value 'PAR' as the source name and a standard error message field (which needs checking on its return). The returned trip id will be used to update the SCH_TRIP record with all of the values extracted into the PAR_XML_TRP record including updating the ROUTE_CODE to the extracted trip id (other fields needing updating include Carrier_Id, Owning Depot, Cost Centre).

For each record on the corresponding PAR_XML_TRIP_STOP (link using FILE_NAME and TRIP_ID) then create a corresponding SCH_TRIP_STOP (STS) record (using the generated MTS trip id, Stop sequence, Stop Type, Location ID and planned Arrival and departure times) and then update it using all of the other extracted values (including kms from previous stop).

NB) The trailer type provided by Paragon in <TRIP_TRAILER_TYPE> needs decoding via ?PARAGON_TRAILERS? (call IMP.Decode_Value) and then used to update STS.TRAILER_TYPE.

For each record on the corresponding PAR_XML_TRIP_ORD (link using the PAR_XML_TRIP_STOP_SEQ value) validate that the order exists in MTS with the correct status in the correct location and create a SCH_HAULAGE_ACTIVITY (SHA) record.

The STOP_LOC_TYPE from PAR_XML_TRIP_STOP (SU, PK, DL or CL) will be used to determine ACTIVITY_NAME (Load or Unload). At the same time update the CURRENT_DEPOT on SCH_ORD to

Existing Trip :-

Extra validation on trip status, any debrief against trip etc. Also check if any data at trip level is being changed (Haulier, Cost Centre etc.)

Update SCH_TRIP (if required).

For each record on PAR_XML_TRIP_STOP (PXTS, link using FILE_NAME and TRIP_ID) get the corresponding SCH_TRIP_STOP (STS) record (link using SCHED_NAME, TRIP_ID and STOP_NO) and compare STOP_LOC with the LOCATION_ID

If matching location (same stop) then :-

Extra validation is required on the stop for any debrief, any fields being changed (Prev distance, planned times etc.).

Update STS (if required).

Delete all of the existing SCH_HAULAGE_ACITIVITY (SHA) record associated to the stop,For each record on PAR_XML_TRIP_ORD (PXTO) then create corresponding SHA record.

If not a matching location (stops moved) then :-

Extra validation on all SHA (any debrief on corresponding orders, order status, current_depot etc.)

Delete all SHA?s

Check if the original location is still on the trip but has been moved to a later Time

If moved then just add the create the new STS in the current position

If deleted then create the new STS and delete all of the following STS records.

For each record on PXTO create SHA records

If any more stops exist on STS than exist on PXTS then delete the extra STS.

If any more PXTS records exists then STS then create the extra ones.

NB) Translate the OMS Ref provided by Paragon by removing the ?-n? from the end as this was added to represent the journey legs.

NB) Extra validation on orders also includes checking that the order exists in correct location with correct status etc. and if updating a trip stop ensure that data will not conflict with any actual times already on file (include break times). Also ensure that every order has a Load and Unload for each trip.

NB) The activity for the orders at the stop (Load/Unload) will be determined by using the stop type. (SU and PK are loads and DL and CL are unloads).

Run standard TRM.VALIDATE_TRIP ??? (Probably not as could delete the SU and CL hub locations and any empty stops).

NB) The following code that is embedded in validate trip routine now needs running stand alone if the standard code is not being run.

Once all of the orders are added to the trip then the ST.SVC_TYPE (Delivery Type) field needs to be populated on the trip based upon the first order (So.Delivery_Type_ID) associated with the trip.

Loop through all of the stops and calculate the time between the planned departing of the previous stop and planned arrival at the current stop and store it in drive_time_from_prev_stop

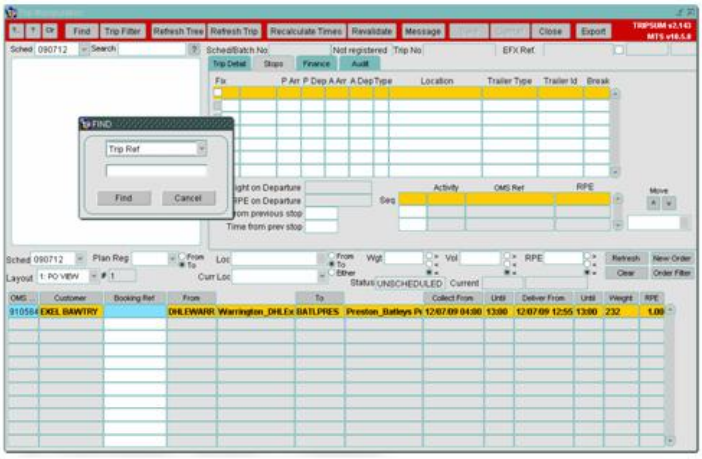
The RPE and weight values on departure from each of the stops can also be calculated and stored in the corresponding fields on STS (weight_on_depart, volume_on_depart, rpe_on_depart, rpe_on_depart_round and du_qty_on_depart). This calculation can be copied from the already existing code in validate trip.

Update the record on PXT for the file to say it was a SUCCESS.

Once all trips are processed from the file then reset the status on all Orders (From NEW to UNSCHEDULED, SCHEDULED or SCHED_COLL).

4.4 Trip Forms

The trip forms (TRIP_PLAN and TRIPSUM) need a new option added to the FIND function.



It will be called Route Code and it will be used to compare against the SCH_TRIP.ROUTE_CODE field.

5 REFERENCES

Not Applicable

6 DOCUMENT HISTORY

Version	Date	Status	Reason	Initials
0.1	08/07/09	Draft	Initial version	DRM
1.0	06/11/09	Issue	Reviewed and Issued	MJC

7 AUTHORISED BY

Matt Crisford	Development Manager
Peter Greer	TMSCC MTS Product Manager