DQIAGILITY RELEASE NOTES

Subject: DQIAgility v7.1.30

JULY 6, 2018

DQIAgility v7.1.30

New version available in downloads – DQIAgility vr7.1.30 (20180418)

Changes:

1. Majure writeback – new option to use the trip_tbl.RouteName instead of the routeheader_tbl.RouteName in the LOAD_ID tag
    a. Also, the routeId is parsed from the trip_tbl.RouteName field before it is written to the LOAD_ID tab. For example, the routeId, 1700 is extracted from the routename 'WL_1700_20180418'

DQIAgility v7.1.29

Changes:

1. Route API – route Ids in Agility are NOT unique per location, so now we need to append the location (branch) to the route name that we store in the trip_tbl to make it unique per location.

DQIAgility v7.1.28

Changes:

1. RouteOrderAPI - The date that is used to format the routename is now used to update the tripDate on the trip record. This would only occur once, when the trip routeName is initially set.

DQIAgility v7.1.27

Changes:

1. Routing with API:
    a. The route Id may either be found in the shipments display API call or the pick files display API call. Previously, it was only looking in the shipments display but now it looks in the pick files display first, then if not found, looks in the shipments display.
    b. Some order that have route Id may not actually need to be route. i.e., Will Call orders. So now, there is an option to exclude orders from routing based on their ship via.
        i. Found under Priority tab -> API Routing tab

DQIAgility v7.1.26

Changes:

1. API sessions appear to be expiring. After logging into the API, a sessionId is returned and stored in memory so that it can be used for subsequent calls. However, that sessionId only lasts X number of minutes before expiring. After that time, any additional calls to the API will fail with a message like 'Login is required to access this application' or 'Session has expired'. To try and prevent this from happening a couple of changes have been made:
    a. Reset the sessionId every time the importer runs a new task (ImportOrder, SyncTables, etc.). This will force a new login to get a new sessionId.
    b. Set the sessionId expire timeout to 2 hours. After that time has elapsed, then log in again to get a new sessionId

c. After each API call, check for login failed messages and if encountered, log in again to get a new sessionId and then retry the API call again up to 5 times. After 5 times, give up.

DQIAgility v7.1.25

Changes:

1. For routing based on the API, some changes have been made:
    a. The RouteName that is saved to the trip_tbl is now formatted as RouteId_ImportDate+1. ImportDate+1 means it's the dTimeStamp date of the order record plus 1 business day.
    b. Resequencing orders based off StopNum. The change is that previously we were calling one query to get all of the orders on the route. However, there isn't way to query all the orders on the route, so now we query each order individually. It's probably more inefficient but seems to be the only way to do it.

DQIAgility v7.1.23

Changes:

1. Bug fix: The BranchId was not being found when a transformed branch was not set up for a branch. It should be able to find it regardless, so that has been fixed and should return the BranchId now.

DQIAgility v7.1.22

Changes:

1. If the RouteId from the soap api returns a blank RouteId, then it should ignore it just return a blank. Previously, it would append the date to a blank RouteId which in the end won't be unique.
2. The BranchId (setup in POS connection) is now used when calling the SOAP API just like it is when calling the REST API.

DQIAgility v7.1.21

Changes:

1. Under the POS Connection form, a new column for Branch Id was added. The Branch Id, which is different than the Branch, is required for API writeback.

DQIAgility v7.1.20

Changes:

1. Use the 'Ship To' field to always set the order address name which should get set to the order_tbl.ShipTo and trip_tbl.ShipTo fields.

DQIAgility v7.1.19

Changes:

1. Changed timestamp delay to go back an extra two minutes for the Main and DeliveryCost writeback routines. Order writebacks could get skipped because the query only writes back order's whose tripTimeIn is greater than 2 minutes, so there could be a 2-minute gap where orders were recently returned but not eligible to write back. But now, the query will go back an extra 2 minutes to cover that gap.

DQIAgility v7.1.18

Changes:

1. Lat/Long Bug fix revised: The previous bug fix for inserting/updating the job lat/long has been revised:
    a. The job lat/long will only be **inserted** if the job is a NoLatLong job
    b. The job lat/long will only be **updated** if the job is not a NoLatLong job and the customer is not a NoJobLatLong customer.
        i. If the option, 'Always update Lat/Long on job account' flag is turned on then the lat/long from Agility will overwrite the lat/long in DQ
        ii. If this option is not turned on, then it will only write the lat/long from Agility if the current job lat/long is currently set.

DQIAgility v7.1.17

Changes:

1. Bug fix: If they send us a lat/long on the order, and the order is a NoLatLong order, then the lat/long was being saved to the customerJobAccount_tbl in addition to the order_tbl. This was causing issues because the job account is a NoLatLong job account. Now, only if the option to 'Always update Lat/Long on job account' is turned on in the imported, will it update the job account lat/long.

DQIAgility v7.1.16

***** db script 20171208 or newer required *****

Changes:

1. Soap API – the URL, username and password can be set on the POS Connection -> Soap API tab.
2. Soap API Routing – use the route from the soap API to combine orders together
    a. Found under Options -> Priority tab -> API Routing tab -> Use routing
    b. After the order has been imported, a call will be made to the soap API to get the routeId for the order. The routeId itself does not represent a trip, but together with the exp delivery date, it does, so a combination of routeId and ExpDeliveryDate is used as the routeName that will represent a unique trip. The routeName is stored in a new field in the trip_tbl called RouteName. If there is an existing trip with the same route name, the order will be combined to that trip. If there is not, then the order's current trip will be set to the routeName and the order will be the first order on the route. If the order is locked, already on a trip with a tripTimeIn or already on a trip that has a routeName already, then it will not be routed again.

c. Load Sequence: Each time, after an order has been routed, all the orders on the route will be queried from the API and sorted by StopNum. The load sequence for the orders on the trip will be re-set based on the StopNum sequence. If there is an order on the trip that is not routed from Agility, it will not be returned in the query and thus will not have a stop number. These orders will be given load sequences that moves them to the end of the trip.
d. Currently there is no way to un-route or change the route for an order. Further work and/or further information will need to be provided from Agility to do this.

DQIAgility v7.1.15

Changes:

1. The Agility API cannot accept an image of type png, so now it is converted to a jpg image before sending it.

DQIAgility v7.1.11

Changes:

1. Main, Delivery Cost and Majure writebacks now triggered by audit table status changes.

DQIAgility v7.1.10

***** requires db update 20170630 or newer *****

Changes:

1. Majure writeback option 'Write files to sub folders based on Branch'
    a. If selected, a sub folder with the branch name will be created under the Majure writeback folder and the files will be placed here
    b. the batch file that writes the majure files to the majure servers will need to be modified to look in each sub folder for the files and move them to the correct server.
2. Agility REST API Writeback:
    a. Connection info for the REST API can be set under the POS Connections individual connections form.
    b. Signature writeback – write back signature file based on order audit status change
        i. Only tran type of SO or CR are written back
    c. Messages writeback – write back the order_tbl.reference field based on order audit status change
        i. Only tran type os CM, SR, DP, PO, Quot, RM, RM-input, RM-output, RM-operation, SO, SPO and WO are written back

PIAgility v7.1.1

Changes:

1. Multiple branches per POS Connection can be entered.
    a. The name of each branch should be entered.
    b. If a branch name needs to be transformed to a different name, then enter that under the 'Transformed Branch' column.

PIAgility v7.1.0

This is the Agility importer converted from vb6 to .NET.

***** requires db update ******

Features:

1. Written with .NET Framework 4.5.2
2. It can import from multiple Agility web services into 1 DQ database.
    a. The Agility web services are uniquely identified using the Branch field set under POS Connection Info.
    b. If the Branch is not unique per POS, then it can be converted to a different branch name using the Transformed Branch field.
        i. As soon as the data is retrieved from the web service, the branch name will be changed to the Transformed Branch name in the Primary Key and Location fields.
        ii. When a transformed branch is written back to the web service, it will be converted back to the original Branch name.
    c. Right now, only 1 DQ database connection is allowed per POS. However, if the need arises, it's possible that the POS connections could be broken out based on the unique Branch Name. However, one could just set up a separate POS to handle this.
    d. The calls to the multiple web services to get new orders is done asynchronously, to minimize waiting time. However, the importer still waits for all the calls to return before they are actually imported.
    e. For order writeback, each order needs to be written back to the web service from which it originated. The PKImport contains the Branch/Transformed Branch name which can be used to find the originating web service.
    f. For table sync writeback, the tables will be written back to ALL POS connections since it's assumed that they all need to stay in sync with the DQ tables.
3. All other settings/features should have been moved over from the old importer, in addition to some new features, like Generate Order and mapzone mapping.
4. To do: DMSI has a new API to write back signatures and messages. I'm not sure if there's an immediate need for this but it can be added to the importer at a future time.
5. And mapzone mapping.
6. To do: DMSI has a new API to write back signatures and messages. I'm not sure if there's an immediate need for this but it can be added to the importer at a future time.