Diagnose and Fix Transportation Data

Updated: 01-05-2004 Supercedes: 05-07-2003

Diagnose.exe is a standalone in-house utility designed to determine whether a client data set has one or more of a variety of data problems. This diagnostic utility checks, and reports on the majority of the primary data files located in the \Server\Dyn\ directory. As its name indicates, Diagnose does not fix any data problems, but does output fairly precise messages and report files to point the user in the right direction towards cleaning up the data set being evaluated. Other applications, like **FixDiagnose.exe**, are the counterparts to **Diagnose.exe**, which you will use in concert to perform the actual data reparations.

As of Diagnose.exe version 7.8 or newer, the Diagnose report files are generated and output to a date-specific subdirectory of the \TMP\ directory specified in the user's ELTransService.cfg file. This path is specified within the startup batch itself, which maintains the following standard format:

Start C:\Elt\Exe\Emu\Diagnose.exe C:\users\UserName\ELTransService.cfg

The above configuration specifies that the utility will diagnose all of the files, and is the default that most people will run. If using the default configuration for Diagnose.exe, you may run the utility from within EMU as a general batch configuration (be sure to update utils.inf).

Any report files that would otherwise have no contents (i.e. 0kb in size) are not created, thus reducing clutter in the output directory. Similarly, the diagnose.rpt contents, when using this standard configuration, include only those section lines with one or more records related to the individual line's information or problem. Any lines that would otherwise have zero records indicated, will be omitted from the report, potentially leaving only section name headers in place.

If you wish to view all section contents (do not omit zero-record lines – like versions prior to 7.8 still show), you must add a '-o' (lower case o) switch to the end of the command line arguments (after config file path). This configuration cannot be run from EMU as a general batch. For example:

Start C:\Elt\Exe\Emu\Diagnose.exe C:\Users\UserName\ELTransService.cfg -o

If you need to run Diagnose with specific flags set to check particular file groups, but not the entire data set, you must set each of fifteen (15) individual flag values at the end of the standalone batch configuration. This configuration also cannot be run from EMU as a general batch. For example:

Start C:\Elt\Exe\Emu\Diagnose.exe C:\Users\UserName\ELTransService.cfg 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Each flag value is either set to off (OFF flag = 0) or on (ON flag = 1) to include or exclude the associated file in the diagnose. The flag positions are (in order) associated with: 1) Student, 2) Stutrn, 3) Stop, 4) Stptrn, 5) Run, 6) Runtrn, 7) Route, 8) Rtetrn, 9) Request, 10) Cmndst, 11) School, 12) Schtrn, 13) Rundir, 14) Cluster, 15) Clustrn.

Note: You cannot combine the '-o' switch option with the individual flag setting option. The utility will not run.

The date-specific subdirectory, where each set of diagnostic reports ends up after running **Diagnose.exe**, is based on the day the reports are generated. The directory name will be "DIAG" plus an 8-digit date value entered in the format "YYYYMMDD." For example, you run Diagnose on May 17, 2001. The output directory name would be: C:\Users\Username\TMP\DIAG20010517\. Any diagnostic reports generated during that same day will be output to the very same directory. If you want to save the set of reports between each run, be sure to copy them to some other location before running Diagnose again. Note also that Diagnose does not delete old folders.

DIAGNOSTICS:

Special note: To obtain the correct sequence in which all 'fix' utilities should be run, refer to the documentation for ELFixApps. This document (ELFixApps.doc) will contain the correct order in which the majority of the fix utilities must normally be run. If you find that a utility indicated in the Fix actions below does not appear in the ELFixApps documentation, please consult Programming before taking further action.

Option #0: Files

Seven standard values are output to DIAGNOSE.RPT, and one report file is generated during this operation.

Output: Number of data files with the wrong version (FileFacts.rpt).
Number of corrupted key files (FileFacts.rpt).
Number of missing key files (FileFacts.rpt).
Number of data files that are the wrong size (FileFacts.rpt).
Number of user files that do not exist. (FileFacts.rpt).
Number of user files that are the wrong size (FileFacts.rpt).
Number of user files that exist but are not used (FileFacts.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

FileFacts.rpt: Lists the particular file that the problem occurs for, and in the cases of files being the wrong size it lists the existing file size and then the size it is supposed to be. In the case of files built a long time ago, there is a good chance that the existing file will be reported as being one block too small. This is ok. If the situation really worries someone, then the file can be extended by a few records thus forcing it to be rebuild with the correct size. If the difference in the file sizes is more than 1, then that indicates a much more serious problem, and should not be handled by extending the file. Please inform programming if such is the case.
Fix: Programming should be informed if any of these situations exist in the data.

Option #1: Student Records (STUDNT):

Seven standard values are output to DIAGNOSE.RPT, and Seven report files are generated during this operation. There are two additional values (and one associated report) output for those data sets with SPED feasibility checking in place.

Output: Number of mismatched student records (Student.rpt).
Number of mismatched student requests (Stureq.rpt).
Number of invalid eligibility codes (Stuelg.rpt).
Number of student records with invalid District ID's (District.rpt).
Number of missing common destination records (stucmn.rpt).
Number of student records with invalid map zone field (studntMzn.rpt).
Number of students with an invalid sch/grd/prg combo for transportation (SchGrdPrg.rpt).
Number of students with missing default trips (MissingTrips.rpt).
Number of students with shuttle timing problems (ShuttleTiming.rpt).

The following are only checked if SPED feasibility checking is active, and ELMaster.dat has SPED on:
Number of SpEd students assigned to RegEd stops (SpEdFlag.rpt).
Number of RegEd students assigned to SpEd stops (SpEdFlag.rpt).

Note: If a client does not want SPED feasibility checking in place, they need to be sure their sysdef.inf file has the SPED setting set to zero (i.e. SPED=0).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Student.rpt: Reports each instance where the student's ID does not match the Record ID.
Fix: **Run FixStudentID.exe**

Stureq.rpt: Lists each request ID that does not point to the correct student record.
Fix: **Run FixStudentPointers.exe** (version 1.6 or newer).

Stuelg.rpt: Reports each student record with an invalid eligibility code (of 8224).
Fix: None yet – contact Programming.

District.rpt: Lists all student records with District ID's that are blank or all zeros.
Fix: Client will handle on-site.

Stucmn.rpt: Lists all student records where cmndst does not match student.
Fix: Run **FixDataCmndst.exe**.

StudntMzn.rpt: Lists all student records where tenth character of map zone is not a space.
Fix: Run **FixMapZoneLength.exe.**

SchGrdPrg.rpt: Lists edulog ID of all students with invalid sch/grd/prg combo for transportation purposes.
Fix: Client will need to resolve on-site. Should be done right away if dependent stutrn, request, and/or cmndst data conditions exist because of these invalid student records.

SpEdFlag.rpt: Lists all students assigned to a stop with a mismatched SpEd designation.
Fix: Run **FixSpedStopFlag.exe**.

MissingTrips.rpt: Lists all student records where the student is missing default trips, including the shuttle ones.
Fix: Run **FillInTrips.exe**.

ShuttleTiming.rpt: Lists all student records where the student gets dropped off at the shuttle location after he/she is supposed to get picked up. Fix: Not yet – contact Programming.

Option #2: Student Transportation Records (STUTRN):

Thirty-six values are output to DIAGNOSE.RPT, and sixteen report files are generated during this operation.

Output:
- Number of invalid services (Stutrn.rpt).
- Number of invalid type 1 trips (NoSchl.rpt).
- Number of invalid type 2 trips (NoSchl.rpt).
- Number of invalid type 3 trips (NoSchl.rpt).
- Number of invalid type 4 trips (NoSchl.rpt).
- Number of invalid trip types (BadType.rpt).
- Number of trips with frequency not fully serviced by run(s) (StutrnFreq.rpt).
- Number of stutrn records with invalid map zone (StutrnMzn.rpt).
- Number of stutrn records with invalid locations (Location.rpt).
- Number of stutrn records with end/start dates different from sysdef (StutrnDte.rpt).
- Number of stutrn records with no school service assigned (NoSchSrv.rpt).
- Number of stutrn records with student assigned to the wrong school service (BTSchSrv.rpt).
- Number of stutrn records with student assigned to the wrong stop service (BTNonSchSrv.rpt).
- Number of stutrn records w/ student school & assigned school service unmatched(WrongSch.rpt)
- Number of stutrn records w/ non-sch service assignment & trip type unmatched (WrongType.rpt)
- Number of stutrn records with invalid windows (TripWnd.rpt)
- Number of duplicated keys (StutrnDup.rpt)
- Number of missing keys (StutrnKeysMissing.rpt)

Errors reported in tripptr.rpt:
- Number of Stutrn pointers less than or equal to zero
- Number of Mismatched stutrn pointers
- Number of Student pointers less than or equal to zero
- Number of Non-existent Student records
- Number of Deleted student records
- Number of Bad student pointers
- Number of Request pointers less than or equal to zero
- Number of Non-existent Request records
- Number of Bad request trip ID's
- Number of Mismatched Head pointers
- Number of Mismatched Tail pointers

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Stutrn.rpt: If the stop service doesn't exist, this file will contain the student record number, trip record number, stop service type(PU/DO), and stop service ID.
Fix: Create them manually from within stops tabular. Make sure the service created exactly matches the service indicated in the trip record (i.e. 8:00AM service 100.123001 for trip must be created as stop service 100.123001 at 8:00AM in the stop record.

NoSchl.rpt: Reports trips that are created without a school service.
Fix: Run **DeleteTripsNoSchool.exe**

BadType.rpt: Reports the stutrn record that has an invalid trip type (1 through 5, 11, and 12 are valid)
Fix: If there are less than a few dozen, note trip assignments as applicable, and then flag each student record for deletion, re-activate the record, re-assign trips to stops as applicable, and then confirm the record out to keyfinder. If there are more than several dozen, contact programming.

StutrnFreq.rpt: Lists stutrn record, student record, trip frequency, and run frequency (which cannot fully cover trip frequency).
Fix: Client to fix on-site by verifying that student trips are assigned to stop services which are on runs that have sufficient frequency to get students to school on all days the student trip frequency is effective. Example: trip is MTW-F, and is assigned to stop service w/ freq. MTWUF, but stop service is only assigned to one run w/ freq. MTW--. In this case, the student is not transported on Fridays, and will appear in this report.

StutrnMzn.rpt: Lists stutrn records with invalid tenth character in map zone.
Fix: Run **FixMapzoneLength.exe**.

Location.rpt: Lists stutrn records with student record for all trips that do not have a location.
Fix: Run **DelTripInvalidLoc.exe**.

StutrnDte.rpt: Lists stutrn records with student record with end/start dates different from sysdef.inf.
Fix: Run **FixDates.exe**, and then rebuild all keys.

NoSchSrv.rpt: Lists stutrn ID for trips that do not have a school service assigned
Fix: Run **DeleteTripsNoSchool.exe**

BTSchSrv.rpt: Lists stutrn ID for trips assigned to the wrong school service time.
Fix: Run **FixTripSchBT.exe**. Make sure to move BTSchSrv.rpt into the \users\username\TMP\ directory before running the utility, as this report acts as the input file for the utility. If FixTripSchBt.exe does not fix all the errors you can try to run with an argument of 1 as this will try to fix trips that are assigned before running with this option please see EdulogUtilities.doc.

Btnonschsrv.rpt: Lists stutrn ID for trips assigned to the wrong non-school service time.
Fix: **FixTripBadService.exe**. Make sure to move BTNonSchSrv.rpt into the \users\username\TMP\ directory before running the utility, as this reports ascts as the input file for the utility. This utility will not work if school service of the trip is incorrect so it is important to first fix all the BTSchSrv.rpt problems.

WrongSch.rpt: Lists stutrn ID for trips with school service assigned incorrectly – not the student's normal valid school or transport school. Fix: Run **FixTripBadSch.exe**. Make sure to move WrongSch.rpt into the \users\username\TMP\ directory before running the utility, as this report acts as the input file for the utility. FixTripBadSch.exe will only purge trips that are not assigned so if you want to purge the trips that are assigned you need to use an argument of one. Please see EdulogUtilities.doc.

Wrongtype.rpt: Lists stutrn ID and student ID of each instance where the trip type does not match the non-school service to which the trip is assigned (i.e. trip types 1&3: type=0, and trip types 2&4: type=1).
Fix: Clients can generally take care of this on-site if there are not that many records indicated in the report. Simply remember trip assignments if applicable, flag the student for deletion, re-activate the record, re-assign trips if applicable, and confirm out of the record. If there are a lot of records indicated in this report, contact Programming for further assistance.

TripWnd.rpt Lists stutrn ID and whether it was the pickup or dropoff window that was wrong, and what values they are. Fix: Run **FixStutrnWindows.exe**. If there are a lot of records indicated in this report, contact Programming for further assistance.

Tripptr.rpt: Lists the stutrn id, and description for all errors found for trip records.
Fix: Save \DYN\ as-is, then run **FixStudentPointers.exe** (version 1.6 or newer). Generally, all eleven (11) items under 'Tripptr.rpt' are fixed by this utility. The only exception will be with "Request pointers less than or equal to zero," which is nothing to worry about. For all others, contact Programming, and give them the copy of \DYN\ that you saved before running the utility.

Stutrndup.rpt: Lists stutrn id's of records whose keys are duplicated.
Fix: Run **Rebldkey.exe**. Please contact programming and let them know that your dataset has this problem.

StutrnKeysMissing.rpt: Lists stutrn id's of records whose keys are not in the stutrn.key file.
Fix: Run **Rebldkey.exe**. Please contact programming and let them know that your dataset has this problem.

Option #3: Stop Records (STOPS):

Seven values are output to TMP:DAIGNOSE.RPT, and seven report files are generated during this operation.

Output:
- Number of duplicated keys (Stopsdup.rpt).
- Number of invalid stops (Stop.rpt).
- Number of stops with mismatched availability flags (Mavail.rpt).
- Number of invalid time1 or time2 flags (Time.rpt).
- Number of invalid home stops (HomeStop.rpt).
- Number of stops with invalid tenth character in map zones (StopMzn.rpt).
- Number of stops with start/end date different from sysdef (StopDte.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Stopsdup.rpt: Contains the Stop ID's of all stops with duplicated keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Stop.rpt: Reports stop records with non-existent schools or clusters.
Fix: Manually delete each listed stop record.

Mavail.rpt: Lists Stops with stptrn records that have mismatched availability flags
Fix: Run **FixStopAvail.exe**.

Time.rpt: Contains the Stop ID's of stops with invalid time1 or time2 flags.
Fix: Run **UpdateTime1and2.exe**. If problem records remain, there still exists some problem between the geocode and the stops. Users can run **BadStp.exe** (from EMU) to get an idea as to what is wrong, and then manually modify the geocode or stops accordingly.

HomeStop.rpt: Contains the Stop ID's of invalid home stops.
Fix: Run **FixHomeStops.exe**.

StopMzn.rpt: Lists Stop ID's of all stops with invalid map zones.
Fix: Run **FixMapzoneLength.exe**.

StopDte.rpt: Contains the stop ID's of the stops with frequency dates that differ from sysdef.inf.
Fix: Run **FixDates.exe**, and then rebuild all keys.

Option #4: Stop Transportation Records (STPTRN):

Eighteen values are output to DIAGNOSE.RPT, and eighteen report files are generated during this operation.

Output:
- Number of stop services with incorrect available frequencies (MavlFreq.rpt).
- Number of stop services with invalid day assignments (BadAvl.rpt).
- Number of stop services where stops do not exist (StpExt.rpt).
- Number of non school stop services where windows are nonzero (WindowsZbt.rpt).
- Number of stop services with start/end date different from sysdef (StptrnDte.rpt).
- Number of duplicated services (DupSrv.rpt and DupSrv.inp).
- Number of transfer services with an invalid position on an assigned run (TrnPos.rpt).
- Number of duplicated keys (StptrnDup.rpt).
- Number of mismatched data and keys (*this will be zero unless Rebuild keys was not run first).
- Number of stop services that do not match bell times (Usrvtme.rpt).
- Number of stop services that have 12:00 AM bell times (StptrnZbt.rpt).
- Number of non-school stop services with 12:00AM bell times and stop loads (StpTrnLod.rpt).
- Number of school stop services that do not match frequency (BTFreq.rpt).
- Number of school stop services that do not exist (NoBtFreq.rpt).
- Number of half assigned transfer services (TrHalf.rpt).
- Number of missing one half of transfer service pair (TrMiss.rpt).
- Number of transfer pairs whose pickup is before drop-off (TransferTimes.rpt).
- Number of transfer pairs with circular run assignments (TransferDest.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

MavlFreq.rpt: Contains all stop services with incorrect available frequency (i.e. the frequency stored in the stptrn file is different from the frequency of the stop minus the days it is assigned to a run), listing Service ID, Stptrn Available Frequency, and the Stop's calculated available frequency.
Fix: Run **FixAvailFreq.exe**.

BadAvl.rpt: Reports all stop services with effective frequencies that doesn't match the days the service is assigned to a run(s). The file lists the Service ID and all Runtrn ID's for the run(s) the service is assigned to.
Fix: None yet – contact Programming.

StpExt.rpt: Outputs the stptrn records that contain invalid or non-existent stops.
Fix: Run **BadStpTrn.exe** (version 1.4 or higher) for non-school and garage services, or DelinvalidSchSrv.exe for school services.

WindowsZbt.rpt: Lists non-school stop services with non-zero windows.
Fix: Run **FixStptrnWindows.exe**.

StptrnDte.rpt: Lists stop service with end/start dates different from sysdef.
Fix: Run **FixDates.exe**, and then rebuild all keys.

DupSrv.rpt: Report file that lists duplicate services by stop service ID.
Fix: Run **FixDates.exe**, and then rebuild all keys. If that does not take care of them all, move DupSrv.inp from \tmp\diag…\ to \tmp\ and run FixDiagnose.exe (option 3 – 6), to delete any duplicate stop service(s) accordingly. For multiple instances of the same duplicate records, re-diagnose and repeat this process as necessary.

DupSrv.inp: An input file that lists duplicate stptrn services.
Fix: None per se. This is the input file for FixDiagnose.exe (option 3 – 6).

TrnPos.rpt: Lists stop service ID's and run ID's of all transfer stops with invalid positions on their assigned runs.
Fix: Client to fix on-site. Re-order services on runs to make them valid. Often, a transfer is positioned before a school service (From-school run) or may exist on run with no destination school stop service to terminate the process. Example: In a To-school scenario, run1 transfers to run2, which has no school stop on it, so students for school coded to that transfer never get dropped off. Or, run two might have a school stop on it, but the position of the school stop is ahead of the transfer, so the run would effectively drop the students off before they are even picked up.

StptrnDup.rpt: Reports on duplicated service keys and service records not matching their respective keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Usrvtme.rpt: Lists stop services ID's, and their invalid bell times, that do not match the correct bell times.
Fix: Evaluate each service individually to determine how to proceed. For any school stop services found, Shift Bell times (from within ETC) for the school's bell time – from the 'good' bell time of the school to the 'bad' service time from the stop service, and then back to the 'good' school bell time again. Shifting bell times may be especially important if there are numerous instances of other associated non-school services that do not match the same school bell time, as it will correctly shift them as well (remember, shifting times will also modify time at stop for services, so beware; if a district manually sets time at stop, this may not be a viable option). Always remember to perform the shift times process FIRST, before continuing with any non-school services. When done shifting times, run **FixDates.exe** and rebuild all keys. Do this BEFORE performing any other tasks! For all non-school stop services, manually delete the service (from within stops tabular) and re-create the service manually, to ensure the service is created correctly (re-assign to runs, etc. as necessary).

Special Note: If you cannot shift times because the same school service time already exists, run **Fixdiagnose.exe** (option 3 – 9) to change the service time to the correct time, and then run **FixDates.exe** and rebuild all keys.

StptrnZbt.rpt: Reports the stop services that have a bell time of 12:00 AM.
Fix: Manually delete the service. If service is a school stop service, run **DelinvalidSchSrv.exe**.

StpTrnLod.rpt: Lists non-school stop service ID's with 12:00AM service times and student loads
Fix: Deassign students from the service and delete the service (from within stops tabular).

BTFreq.rpt: lists school services which have the wrong frequency.
Fix: Run **VerifySchServices.exe**. If this utility does not resolve a service, because the service exactly matches an existing service, run Fixdiagnose.exe (option 2 – 5) to delete the second instance of the service. Then run CleanupDuplicateKeys.exe, and rebuild all keys.

NoBtFreq.rpt: list of school services that could exist, but do not.
Fix: Run **CreateSchoolServices.exe**.

TrHalf.rpt: lists transfer service pairs by ID: assigned half of service first, and unassigned half second.
Fix: Run **FixDiagnose.exe** (option 3 – 8) on the half of the service pair which 'IS' assigned to a run. The utility will delete both services correctly.

TrMiss.rpt: Lists transfer services which do not have a corresponding service pair.
Fix: Run **DelMissingTransfer.exe**. If this does not resolve all instances, run FixDiagnose.exe(option 3 – 8).

Trnsfr.rpt: Lists all existing transfer stops within the data set.
Fix: **No fix needed**. This is informational only.

TransferTimes.rpt: Lists all transfers whose pick up time is before its drop off time. Report includes service ids, run ids and times at stop. Fix: Users will have to manually re-adjust their runs or times at stop. There is no programmatic fix available.

TransferDest.rpt Lists all transfer pairs which never end up going to school. Report includes service ids, run ids, and a list containing the sequence of transfer pairs that end up circular.

Option #5: Run Records (RUNS):

Seventeen values are output to DIAGNOSE.RPT, and sevevteen report files are generated during this operation.

Output:
Number of runs that number stops do not match the stop count (NumStp.rpt).
Number of runs with no stops (NoStop.rpt).
Number of runs with no school stops (NoSchool.rpt).
Number of runs with time at school outside early/late window (Runtme.rpt).
Number of runs that do not match bell times (UnRunTme.rpt).
Number of runs that have a 12:00 AM bell time (RunZbt.rpt).
Number of runs with duplicated services (RunSrvDup.rpt).
Number of runs whose begin and end times do not match stops on run (BegEnd.rpt).
Number of circular runs with invalid destination times (Circular.rpt).
Number of circular runs with transfers assigned (TranCirc.rpt).
Number of runs whose first or last stop isn't a school or a checkpoint (BadFirstLast.rpt).
Number of runs with invalid school or cluster (Run.rpt).
Number of runs with invalid To/From flag (ToFrom.rpt).
Number of runs with invalid Rtetrn records (RunRte.rpt).
Number of duplicated sequence numbers (DupSeq.rpt).
Number of missing sequence numbers (SeqMis.rpt).
Number of runs with start/end date different from sysdef.inf (RunDte.rpt).
Number of duplicated keys (RunsDup.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

NumStp.rpt: Reports all runs where the number of stops stored in the run is different from the actual number of stops on the run(i.e. the number of runtrn records for that run). The output is Run ID, Number of Stops, and Number of Runtrn Records for the given run.
Fix: Run **FixSeq.exe**. If the report indicates a record number but no run ID, and Fixseq.exe does not resolve the problem for a given run record, use FixDiagnose.exe (option 2 – 6) to force the record deletion, then be certain to rebuild all keys immediately thereafter.

NoStop.rpt: Lists any runs which have no stops on them.
Fix: Run **FixSeq.exe** and then run **DeleteRunsNoStops.exe**. For any garage runs that show up in this report, run ResetRtePointers.exe. If any garage or deadhead runs remain unfixed, find out which route(s) they are on and deassign appropriately; then re-assign to get valid garage and/or deadhead runs back in place.

NoSchool.rpt: Lists runs with no correctly coded school stop service on them.
Fix: Run **FixRunsNoSchService.exe**. If the utility does not fix a given run, manually delete the bad run, and then re-create the run.

Runtme.rpt: Contains the Run ID, School Service, time at school, early window and late window, for each run that has an time at school outside the early/late windows. Fix: This is a report that is sent to the client as they will decide if they want to change the timing of the runs. This does not have to be changed but information for the district to decide what they want to do. If they want the times changed they can manually do it with editing time at stops in runs.

UnRuntme.rpt: Reports all deadhead and other non-garage runs that have destination times that don't match a school bell time. Fix: Verify the correct destination time and To/From flag first. If the problem is a wrong To/From flag, manually switch the flag to the correct value. If the destination time is invalid (and cannot be fixed with shifting bell times in schools tabular), then run **FixDiagnose.exe** (option 3 – 10). Be sure to enter the time value in as the number of minutes past midnight (i.e. 7:15AM = 435 min. past midnight).

RunZbt.rpt: Lists all the runs that have a bell time of 12:00 AM.
Fix: Manually delete the runs.

RunSrvDup.rpt: Lists the runs that have duplicated services on them.
Fix: First, run **FixDupSchSrvRun.exe**. If, on the next diagnose, there are still runs indicated, run **FixRunStops.exe** – enter run ID initially, then select stop service from list that comes up on-screen. Enter the 3-digit index number for the 'second' instance of the duplicated service.

BegEnd.rpt Lists Run ID of runs whose begin time does not match the time at stop of the first stop on the run and/or end time does not match the time at stop of the last stop on the run.
Fix: Run **FixRunsEarlyLate.exe**.

Circular.rpt Lists all circular runs, by record number and run ID, with invalid destination times.
Fix: Informational for district – no fix required. If desired, user can change To/From flag to match the direction indicated by the initial destination time of the run.

TranCirc_rpt Lists all circular runs, by record number and run ID, that have transfers assigned. There is no automatic fix for this problem. Users will have to evaluate each run individually, and decide how to fix the problem.

BadFirstLast.rpt Lists all runs whose first (for from-school runs) or last (for to-school runs) stop isn't a school or a checkpoint. Fix: **None yet**. Users will have to evaluate each run individually, and decide if it needs to be modified. Contact programming with questions.

Run.rpt: Lists all runs with invalid or non-existent schools or clusters.
Fix: Manually delete the runs.

ToFrom.rpt: Outputs the Run ID of all runs with invalid To/From flags.
Fix: Evaluate each run to determine what the correct To/From flag should be. Manually change To/From flag as appropriate.

RunRte.rpt: Lists the Run ID of all runs with invalid Rtetrn record values.
Fix: Run **ResetRtePointers.exe**.

DupSeq.rpt: Lists the Run ID and sequence numbers of all duplicate Runtrn records.
Fix: Run **FixRunSeq.exe**.

SeqMis.rpt: Lists Run ID of all Runtrn records with missing sequence numbers.
Fix: Run **FixRunSeq.exe**.

RunDte.rpt: Lists the runs with end/start dates different from sysdef.inf.
Fix: Run **FixDates.exe**, and then rebuild all keys.

RunsDup.rpt: Contains the Run ID's of all instances of duplicated keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Option #6: Run Transportation Records (RUNTRN):

Seven values are output to DIAGNOSE.RPT, and seven report files are generated during this operation.

Output:
Number of invalid runs (FreRun.rpt).
Number of invalid services (Runtrn.rpt).
Number of services with invalid effective frequencies (EffFreq.rpt).
Number of services with no effective frequency (NoEffFreq.rpt).
Number of services where the matching school service is not assigned to the run (BTRuntrn.rpt).
Number of services with time next less then zero (TimeNext.rpt).
Number of invalid services (InvalidService.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

FreRun.rpt: Lists all runtrn records that show runs that do not actually exist.
Fix: Run **BadRuntrn.exe**.

Runtrn.rpt: If the stop service does not exist, the Run ID and invalid Service ID are listed.
Fix: Manually add the stop service(s) in stops tabular, to ensure they are created correctly. If the record indicated is a deadhead run, determine which route it is assigned to, deassign an adjacent

run from the route, and re-assign the run to the route. This will delete and re-create a valid deadhead run. In cases where a stop service is needed, but the stop itself does not exist at all, deassign the service from the run, as the indicated service is completely invalid at this point.

EffFreq.rpt: Lists each instance of a run ID and stop service with invalid effective frequency.
Fix: Run **FixRuntrnEffFreq.exe**.

NoEffFreq.rpt: Lists each instance of a run ID and stop service with missing effective frequency.
Fix: Run **FixRuntrnEffFreq.exe**. If this does not resolve the problem, manually deassign the service from the run, as it is completely invalid for that run.

BTRuntrn.rpt: Lists each instance of runtrn records which do not have the proper destination on the run.
Fix: Manually fix the run(s) – this will depend on what each run looks like; it could be a wrong school service, or the right school service with the wrong bell time. Deassign the service from the run and then re-assign a valid service in its place. If the service you are trying to assign is already assigned to a different run, you must decide which run the service belongs on, and deassign/re-assign the service accordingly.

TimeNext.rpt Lists the run id and stop services which have a time next of less then zero.
Fix: Contact programming as have not decided what needs to be done.

InvalidService.rpt: Lists the run id and runtrn records that have a blank service id.
Fix: Contact programming as have no \t decided what needs to be done.

Option #7: Route Records (ROUTE):

Three values are output DIAGNOSE.RPT, and three report files are generated during this operation.

Output: Number of routes that do not have runs (Route.rpt).
Number of routes with start/end date different from sysdef (RteDte.rpt).
Number of routes without a frequency (RteFreq.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Route.rpt: If the routes does not have runs on it, the route record and route ID are listed.
Fix: Run **DeleteRoutes.exe**, or delete the specified route(s) manually. If district needs to keep these routes active, but empty, they need to realize that this will cause problems with their ability to generate reports containing any links to routes.

RteDte.rpt: Lists the route ID's that have start/end date different from sysdef.inf.
Fix: Run **FixDates.exe**, and then rebuild all keys.

RteFreq.rpt: Lists the Route Id's that do not have a frequency.
Fix: First try to change the frequency to what you want it to be. If previous step does not work then deassign all runs from the and change the frequency of the route and reassign the runs to the route.

Option #8: Route Transportation Records (RTETRN):

Two values are output to TMP:DAIGNOSE.RPT, and two report files are generated during this operation.

Output: Number of rtetrn records where route does not exist (Rtetrn.rpt).
Number of rtetrn records where run does not exist (RteRun.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Rtetrn.rpt: Lists the Rtetrn ID and the corresponding invalid Route ID.
Fix: Run **BadRtetrn.exe**.

RteRun.rpt: Lists the Rtetrn ID and its associated invalid Run ID.
Fix: Run **DeassignInvalidRuns.exe**.

Option #9: Request Records (REQUST):

Twelve values are output to DIAGNOSE.RPT, and two report files are generated during this operation.

Output: Errors reported in RequstErr.rpt
Number of request pointers less than or equal to zero
Number of student pointers less than or equal to zero

Number of trip pointers less than zero
Number of mismatched request pointers
Number of bad student request ID's
Number of bad trip request ID's
Number of requests with trip pointer equal to zero
Number of Non-existent Trip records
Number of Non-existent Student records
Number of Deleted student records
Number of Mismatched Head pointers
Number of Mismatched Tail pointers

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

DiaReq.aud: Lists a statistical summary of request records that are: empty, deleted , singletons, or non-singletons. It also includes an informational numeric summary with a total count of errors found.
Fix: **No fix needed**. This is informational only.

RequstErr.rpt: Lists the Request ID, error code, and description for all errors found.
Fix: Save \DYN\ as-is, then run **FixStudentPointers.exe** (version 1.6 or newer). Generally, all twelve items under 'RequstErr.rpt' are fixed by this. The only exception might be with "Request pointers less than or equal to zero," which is nothing to worry about. For all others, contact Programming, and give them the copy of \DYN\ that you saved before running the utility.

Option #10: Common Destination Records (CMNDST):
Five values are output to DIAGNOSE.RPT, and four report files are generated during this operation.

Output:
Number of Cmndst records with non-existent or deleted students (Cmndst.rpt).
Number of Cmndst records with mismatched head pointers (CmnPtr.rpt).
Number of Cmndst records with mismatched tail pointers (CmnPtr.rpt).
Number of Cmndst records where school is invalid for student (CmndstSch.rpt).
Number of Cmndst records with invalid tenth character in map zones (CmndstMzn.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Cmndst.rpt: This file lists cmndst records where the student does not exist or is deleted.
Fix: Run **FixDataCmndst.exe**.

CmnPtr.rpt: Reports each cmndst record that has mismatched head or tail pointers.
Fix: Run **FixDataCmndst.exe**. If this does not resolve all records, use recorddump.exe to verify that all cmndst records indicated in the report are indeed invalid, and have no other information pointing back to them. If this is the case, run **Fixdiagnose.exe** (option 2 – 16) to delete each remaining bad record.

CmndstSch.rpt: Lists cmndst and student record where student's cmndst contains an invalid school entry.
Fix: Run **FixDataCmndst.exe**.

CmndstMzn.rpt: Lists cmndst records containing invalid map zone field contents.
Fix: Run **FixMapzoneLength.exe**.

Option #11: School Records (SCHOLS):
Four values are output to DIAGNOSE.RPT, and four report files are generated during this operation.

Output:
Number of schools with no attendance boundaries posted (SchBnd.rpt).
Number of schools without school stop (SchStop.rpt).
Number of schools with invalid tenth character in map zones (SchoolMzn.rpt).
Number of duplicated keys (SchoolDup.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

SchBnd.rpt: This file lists school records with no attendance boundaries posted.
Fix: Client will handle on-site.

SchStop.rpt: Lists the school code and record for schools with a school stop.

Fix: Run **FixSchoolStops.exe**.

SchoolMzn.rpt: Lists the school record and code for invalid map zones.
Fix: Run **FixMapzoneLength.exe**.

SchoolDup.rpt: Lists duplicated school keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Option #12: School Transportation Records (SCHTRN):
Eight values are output to DIAGNOSE.RPT, and seven report files are generated during this operation.

Output:
Number of duplicated keys (SchtrnDup.rpt).
Number of invalid schtrn records (Schtrn.rpt).
Number of schtrn records with invalid schools (School.rpt).
Number of schtrn records with 12:00AM bell times (Midnite.rpt).
Number of schtrn records with 12:00AM windows (Windows.rpt).
Number of schtrn records with late bell time before early (EarlyLate.rpt).
Number of schtrn records with invalid eligibility type (EligType.rpt).
Number of schtrn records with invalid hazard level (hazardlevel.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

SchtrnDup.rpt: Lists duplicated schtrn keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Schtrn.rpt: Outputs the Schtrn ID that corresponds to records with any non-ASCII data in the school, grade, or program fields.
Fix: Run **FixAscii.exe**.

School.rpt: Lists school transportation records with invalid or non-existent schools.
Fix: None yet – contact Programming.

Midnite.rpt: Lists each instance of schtrn records, by school ID, with 12:00AM bell times.
Fix: No fix required, as the report just indicates which schtrn records do not have bell times associated with them. Clients will usually handle on-site as needed, if they see a problem.

Windows.rpt: Lists each instance of sch/grd/prg with 12:00AM early/late arrival windows.
Fix: Shift Bell Times from within Schools Tabular to add valid windows.

EarlyLate.rpt: Lists each school/grade/program combo with the late bell time before the early bell time.
Fix: Shift Bell Times from within Schools Tabular to reverse times correctly.

EligType.rpt: Lists each school/grade/program combo with an invalid eligibility type of 0.
Fix: District to fix by changing the eligibility type to something valid in the system or removing transportation attached including bell times.

HazardLevel.rpt: Lists each school/grade/program combo with an invalid hazard level of less then 1 or greater than 4. Fix: District to fix by changing the hazard level to something valid in the system or removing transportation attached including bell times.

Option #13: Run Direction Records (RUNDIR):
Eleven values are output to DIAGNOSE.RPT, and eleven report files are generated during this operation.

Output:
Number of rundir records with invalid Rundir information (RunDir.rpt).
Number of rundir records that have only one stop – informational only (OneStop.rpt).
Number of rundir records with 2 gaps next to each other (TwoGaps.rpt).
Number of rundir records with invalid run ID's (BadRunID.rpt).
Number of rundir records with invalid service ID's (BadFStp.rpt).
Number of rundir records with missing services (MsgStp.rpt).
Number of rundir records with invalid stops on run (NotOnRun.rpt).
Number of rundir records with invalid extension record pointers (BadPtr.rpt).
Number of rundir records w/ missing segment ID before & after segment stops (MsgSegment.rpt)
Number of rundir records with duplicated keys (RunDirDup.rpt).
Number of rundir records with overlapping covers for run (Cover.rpt).
Number of rundir records with invalid stops on cover (Notoncover.rpt).
Number of rundir records with segments at beginning (beginsegment.rpt).
Number of rundir records with segments at end (endsegment.rpt).

Number of rundir records with incorrect deadheads (DeadHead.rpt).
Number of rundir records with incorrect route information (RundirRoute.rpt).
Number of rundir records with incorrect frequency (RouteCovers.rpt).
Number of rundir records with bad nodes (RundirBadNodes.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

RunDir.rpt: Lists the records containing invalid Rundir information.
Fix: Run **FixBadRundir.exe**.

TwoGaps.rpt: Lists rundir records which, if displayed in graphical run directions, would generate output to screen indicating the error for 'Two gaps next to each other.'
Fix: Run **FixBadRundir.exe**.

BadRunID.rpt: Lists rundir records with Run ID that does not exist.
Fix: Run **FixBadRundir.exe** (will purge the record, as it is unrecoverable).

BadFStp.rpt: Lists rundir records which contain stop service ID's that do not exist.
Fix: Run **FixBadRundir.exe** (will attempt to locate and assign the correct service).

MsgStp.rpt: Lists rundir records that are missing one or more stop services.
Fix: Run **FixBadRundir.exe** (will attempt to insert the missing services into the run directions).

NotOnRun.rpt: Lists rundir records that include services which are not actually on the run anymore.
Fix: Run **FixBadRundir.exe** (will attempt to remove invalid services from the run directions).

BadPtr.rpt: Lists rundir records that have invalid extension records.
Fix: Run **FixBadRundir.exe** (will eliminate the bad extension records, and try to re-construct valid run directions).

MsgSegment.rpt: Lists rundir record number and run ID of runs with a missing segment ID before and after segment stops.
Fix: Run **FixBadRundir.exe** (will insert ID of missing segment before/after problem stop).

RunDirDup.rpt: Lists duplicated rundir keys.
Fix: Run **CleanupDuplicateKeys.exe**.

Cover.rpt: List rundir record number and run ID of runs with overlapping cover.
Fix: Run **FixBadRundir.exe**

Notoncover.rpt: Lists rundir records which contain invalid stops in them.
Fix: Run **FixBadRundir.exe**

Beginsegment.rpt: Lists rundir records which are invalid because the run directions start with a segment instead of a stop. Fix: Run **FixBadRundir.exe**

Endsegment.rpt: Lists rundir records which are invalid because the run directions end with a segment instead of a stop. Fix: Run **FixBadRundir.exe**

DeadHead.rpt: Lists the records which contain invalid deadhead information. It may be that the deadhead is wrong, missing or is there when it shouldn't be.
Fix: Run **FixBadRundir.exe**.

RundirRoute.rpt: Lists the records which contain invalid route information. It may be that the route information is wrong, missing or is there when it shouldn't be.
Fix: Run **FixBadRundir.exe**.

RouteCovers.rpt: Lists the records which contain invalid cover information. The cover in the rundir record is not a valid cover for the run.
Fix: Please contact programming if you get this error.

RundirBadNodes.rpt: Lists the records which contain segment records that have a valid mapseg arcid, but invalid nodes.
Fix: Run **FixBadRundir.exe**.

Note: If **FixBadRunDir.exe** does not completely clean up the problem areas after its first run-through, check your user directory's \Tmp**Fixdir.aud** file, which reports any runs with corrupted directions. The audit file contains an action report that distinguishes between which records were fixed and which remain as problematic. Re-run the utility until all that remains in Fixdir.aud are runs for which the action identifier is "could not fix." For these runs, you will need to create a work list including all listed runs. Then go into Runs - Group Processes – Directions, and re-generate driver directions for this list, making sure to check the 'delete existing directions' checkbox before initiating the processes.

Option #14: School Cluster Records (CLUSTER):
Diagnose does not currently check anything for this option.

Output: NONE

Option #15: School Cluster Transportation Records (CLUSTRN):
Three values are output to DIAGNOSE.RPT, and three report files are generated during this operation.

Output: Number of invalid cluster transportation records (Clustrn.rpt).
Number of invalid cluster transportation school times (CSchTme.rpt).
Number of invalid cluster transportation To/From flags (CToFrom.rpt).

Report file output with descriptions of contents, and 'Fix' solutions to resolve the data problems:

Clustrn.rpt: Lists all invalid Clustrn records with non-existent schools or clusters.
Fix: None yet – contact Programming.
CSchTme.rpt: Lists each instance of clustrn records that contain invalid school times.
Fix: None yet – contact Programming.
CToFrom.rpt: Lists the clustrn record numbers containing invalid To/From values.
Fix: None yet – contact Programming.

Important tips on steps to follow when fixing data:

As a general rule, begin fixing data by resolving school, school transportation, and school stop/stop-service problems first. Then you can take care of students, stops, runs, and routes (usually in that order). You will often want to resolve the following areas last: invalid begin/end times for runs, runtrn effective frequency problems, and bad run directions.

One of the very first things that should be done is to create services that are missing. The services that you should create will usually be found in Stutrn.rpt and Runtrn.rpt. Create these services in stops tabular, and make sure they are created correctly (i.e. valid services for valid bell times). It is important to create each service with the correct service ID. For example: The 7:00am run 100.001 indicates stop service 100.222003 is on the run, but the stop service does not exist. When creating the new stop service, be certain that you are able to create the service for the 7:00am bell time, and that the service ID is 100.222003). If you are unable to create the correct service manually, then you must substitute one service for another using FixDiagnose. Please be very careful when creating these services.