REFERENTIAL INTEGRITY FIXES

Data files within STD*MIS are linked together through system-generated id numbers called primary and foreign keys. These 10-11 digit id numbers link pieces of data that together form the picture that is viewed with the application by the user. Unfortunately, and for a variety of reasons, sometimes these keys become lost, causing pieces of data to be come unlinked from others. These are known as referential integrity errors. For example, if a user views an Interview Record and notices that it does not have a diagnosis or case number associated with it, this is a referential integrity error - the case record has become unlinked from the interview record. Case.dbf holds the case number and the diagnosis for a specific case and is linked to the interview record (Ira.dbf and Irb.dbf) by the Irb_id. Correcting these errors can be a very complicated process and knowledge of STD data and the business processes of STD prevention can help in resolving some of the issues involved with referential clean-up. Though STD knowledge can assist in understanding how to fix referential integrity errors, it is more important to understand how the databases relate to one another. The causes of these errors are varied and can be difficult to identify; some possible causes of referential integrity errors include corrupted data, conversion issues, network instability, power outages/surges, computers locking and freezing in the middle of data entry, existing the application incorrectly, and human error. Regardless of how they began, it is important to the functioning of the system to take care of them in a timely manner.

Documents such as the **STD*MIS Data Dictionary**, **Fields List**, and schematics (**Key Linkages)** are invaluable tools for correcting referential integrity errors. The data dictionary outlines the database structure for all of the required databases in STD*MIS, including variable length, type, and description. The **Fields List** document lists all of the variables alphabetically and which databases they are located in. This is a very important document in helping to repair referential integrity problems. The schematic shows the pictorial linkage between the database files. If your site does not have these tools, please consult with your program area's STD*MIS support person. Contact information for STD*MIS support staff and their assigned project areas are available on the STD*MIS website at http://www.cdc.gov/nchstp/dstd/STD-MIS-Contacts.htm .

Anyone attempting to correct referential integrity errors should exercise extreme caution when working with the database files. Changes done at this level are potentially damaging and are usually permanent. It is strongly recommended that a ***copy*** of the actual database should be used to seek out solutions to problems. Solutions should be documented and then implemented in the copy of the database. After the solutions have been verified in the copy by running the referential integrity report again, they can be implemented on the live database. This approach will not only reduce the chance of severe damage being done to the live database, but will also decrease the amount of time the live application must be down. Since referential integrity error correction requires exclusive access to the databases (i.e. no one else can be in the system), the less time spent implementing solutions, the less time data entry and/or clinic operations are suspended. Regardless, project areas must ensure their system is backed up on a routine basis and that back ups are locatable, and useable.

RUNNING THE REFERENTIAL INTEGRITY REPORT

The referential integrity report can be accessed through AMaintenance, System Checks, Database QA reports, Referential Integrity Check.@ Access to the report (and all other Maintenance functions) requires an administrative-level user id and password. This report requires exclusive access to the databases and must be run while no other users are logged into the application. The report will take from a few minutes to an hour or more to run, depending on the size of the individual dbf's. After the data files have been loaded, the user will be prompted to press any key to continue. A list of data files to select will appear. The user may select a single file, multiple files or all of the files. To select all of the files, the user must hold down the <Ctrl> key and press the letter <T> key. A check mark will appear beside the files selected to be reviewed. To select an individual file, use the up and down arrow keys to move to the desired file; when the correct file is highlighted, press the <Space Bar>. To remove an individual file from selection, use the arrow keys to move to the file, highlight it and press the <Space Bar> again. The check mark is removed and the file is removed from the review. Once the user has selected the desired file(s) for review, press the <Enter> key to start the check process. The length of time it takes the report to check the files will depend upon the size of the databases. The more data stored the longer the report will take. Once completed, the errors will be displayed. The user will need to choose AP@ to print the report, either to the printer, a text file or both, once the program has completed running.

Note: Users should periodically delete seekkeys.dbf from the application folder before running the Referential Integrity Check again.

DOCUMENTING CHANGES IN THE DATABASE FILES

When making changes in the database files it is important to keep documentation on those changes. Using a simple grid to document steps taken to rectify referential errors can help guide the user through the clean-up process. With documentation an historical overview of the changes made within the files can be tracked despite changes in personnel and over greater periods of time. This documentation will also aide the data person=s memory as to what changes were made, when, and why. A sample of a grid follows below with a complete copy of the grid attached to this document. User=s may modify this grid or use it as a template to create their own.

For Case.dbf: error Morb_id not in Morbrept:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Morb_id xxxxxxxxxx not in Morbrept	Fr(b)_id	>Fr.Event_id	>Morb.morbid			Replace morb_id w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx			

> - to file.field name xxxxxxxxxx - variable value

When implementing the same resolution for a large number of problems, it may not be necessary to document individual solutions. In the error box, the data manager may wish to document the starting point through the ending point of the records that will be resolved using the same solution. An example is in the case of removing corrupted data. If the corrupted data covers records 25 though 52, the user can simply state: Errors - records 25 - 52; with Solutions - deletion of corrupted records. A problem may take more than one line to resolve. Record numbers from various files should also be documented as well. This documentation may be kept electronically since it is a simple grid and may be manipulated to the user-s benefit. It is also recommended that if any short hand is used, a key should be placed on each page where the short hand is used. This will allow others who may come behind you to understand what was done during data correction.

CORRECTING REFERENTIAL ERRORS

Resolving referential integrity errors often takes time and a bit of determination. Occasionally, implementing solutions may create additional errors which must then be resolved before there can be a reduction in the total number of errors. At times it may seem like a never ending battle, but eventually it will pay off. The following suggestions are by no means the only methods to resolve the problems listed in most cases, nor is this an all encompassing list of solutions. When possible, compare variables, such as dates, and other data to make sure the solution chosen is the best for the problem. Program areas may find systemic or area-specific problems and may need to call their STD*MIS support person for help in resolving them. The user may need to experiment and find methods that may be more suitable to their individual program area. Once any problem is resolved the file that has been changed must be reindexed before the solution can be viewed in the application. There are two ways to reindex files: 1) open a DOS prompt; navigate to your STD*MIS folder (example: C:\std\stdmis34b); type in <STDMIS A> and press <ENTER>; this will automatically launch reindexing, and will reindex the entire system; 2) go to AMaintenance, Reindex System, Data Files;@ use the <Space bar> to select individual files, or select all files by holding down the <Ctrl> key and pressing the letter <T> key. Once again, it is advised that a copy of the system should be used to test fixes before enacting them on the live system.

The person fixing these errors should be aware of local programmatic policy with regard to deletion of data from the system. Sometimes deletion of record(s) is the only solution available to resolve issues. Solutions below begin with the assumption that records are linked at the time of data entry. If records are not linked, it may be necessary to recreate the missing key record. If linked records are marked for deletion, then the user must determine if the record that contains the error should also be marked for deletion or if they should empty the variable field. Also, Patient_ids and Address_ids are not presented as methods for possible resolution for referential integrity problems, though they are found in most databases. Patient and address ids may be used at the data manager-s discretion.

Foxpro Database Utility (FDBU) is bundled with STD*MIS. Fixes suggested below are presented using this database maintenance software. Solutions to referential integrity errors can be completed with other software; however, data managers should be completely aware of the

side effects that may be caused by trying to apply those solutions directly. Some applications will alter the data structure; if the data is then transferred back into STD*MIS, it may create much larger problems and possibly complete loss of data. So while it may be easier to solve a problem in another data management application, the actual solution may have to be implemented through FDBU or another approved tool. Please approach any resolution of database errors with caution.

Locating a record in FDBU

There are two ways to reach FDBU. One way is through the DOS prompt and the other is through Windows. In DOS, type FDBU in the STD*MIS path; for example: <C:\stdmis34\fdbu>, then press <enter>. In Windows Explorer, in the STD*MIS folder, locate and double click on AFDBU.exe.@ Either method will launch the application. A blue window will appear, with commands (“F” keys) listed across the top, and headings for “Files”, “Indexes” and “Fields” down the center. In the grey box appearing under “Files”, type in the name of the file that contains the referential integrity error and press <enter>. If typing the name of the dbf does not work, press the <F2> key and with ADatabase@ highlighted, press Aenter@. This will bring up a menu of the .dbf files in STD*MIS. Type in the first letter of the file name desired. Use the arrow key to scroll down to the file needed. Once the file desired is highlighted, press <enter> twice. The name of the chosen file will appear under “Files”, and the variables in the file will be listed under “Fields”. To view the contents of the file press the <F5> (Browse) key and <enter>. This will open the .dbf file (in turquoise) across the bottom of the window. With the file open, press the <F7> (Move) key. This menu is where search functions are found. If you have the record number, the AGoto@ function can be used. Simply type in the record number and press <enter> twice. To search on a variable, such as event_id or fr_id, the “Locate” function must be used. With ALocate@ highlighted, press <enter>. In the grey pop-up box type in the name of the variable you wish to search on, then an equal sign, and value of the variable in single quote marks. Example: patient_id = ›0110000025'. Press <enter> twice once the statement is complete. Keep in mind that some variable values can be in a file more than once, such as patient_id, event_id, and address_id. To repeat the search with the same variable and value, simply press AF7" and <enter>, <enter> repeatedly, until either you locate the desired record or the message AFile not found@ appears in red, underneath the “F1” (Help) command. The “locate” function begins its search from the current location in the file, so it is always a good idea to return to record one before beginning a new search. To return to record one, press <ctrl> and <page up> at the same time.

More complicated searches can be completed using AF7,@ Alocate.@ Compound statements can be used to help narrow a search. The following example is in Reactor.dbf; the error is Reactor_id not found in Test.dbf. Since both Test and Reactor files have Event_id and Receive_dt as variables, an option to trying to find the missing record may be to search on both variables at the same time. The locate statement would be as follows:

Expression event_id = ›xxxxxxxxxx‹ .and. receive_dt = ctod(›xx/xx/xx‹)

Where xxx’s = numeric values

Operating signs are set off by the use of periods before and after they are written (hence A.and.@). Dates are actually stored differently than they appear so the CTOD phrase is used to convert them into date format as written above with months, days, and years substituted for the Xs. Documentation:

Errors	**Dbf: Reactor Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Reactor_id xxxxxxxxxx not in Test	Receive_dt	>Test - search				Place Reac_id xxxxxxxxxx in Test
	xx/xx/xxxx	Reac_id & Rec_dt				

> - to file.field name xxxxxxxxxx - variable value

This is part of the reason why fixes focusing on Patient_id and Address_id are not fully explored in this document. Both these variables are found in many databases. By using a combination of Patient_id or Address_id and a variable unique to the file that is missing data, many different compound searches can be created and used. Using the same example as directly above, a search could be completed on Patient_id and Receive_dt as well. However, the danger in using Patient_id and Address_id as search tools is that they can appear more than once in a file. So the user should review what is found and make a determination as to which record should be used in any fixes. This is why it is best to pair either of these variables with a unique one, though depending on the file, these variables may repeat as well.

Marking records for deletion and deleting variable values

Marking records for deletion and deleting variable values may be a part of referential error resolution. In fact they may be the only solution available for some problems. However, the user should be very sure that this is the proper course of action before proceeding. Before deleting records, make sure that all other options have been examined. Once it is decided that deletion is the right thing to do, accomplishing the task is very easy. To delete a record, make sure it is highlighted, anywhere within the record is fine, and press the <delete> key. The word ADELETED@ will appear on the top line of the turquoise box, above the third variable name. (Record deletion in STD*MIS is logical deletion, rather then physical. The record is still physically present, but is marked for deletion.) Any time a deleted record is highlighted this message will appear. To undelete the record, follow the same steps and the message will disappear. Deleting the content of a specific variable is almost as easy. In the desired record use the arrow keys until the unwanted variable field is highlighted, then press <enter>. This will activate the field: the highlight becomes white, the data becomes black, and a blinking cursor will appear in the field. Pressing the <delete> key until all of the information is gone will clear the field. Press <enter> again to close the field. If the cursor does not appear after pressing <enter> to activate the field, then the user may have accidentally deleted the entire record. Deletion of records or variable values will not be evident in the application until the affected files are reindexed.

In choosing between deleting a record or a value in a specific variable within a given record, the user must consider whether or not the record itself is valid. Determining whether a record is valid or not requires locating and evaluating any linked records. If all other linked records (use the **Fields List** to find what links are available to a given .dbf) are deleted, then the record being investigated probably needs to be deleted as well. However, if a link is found to the record being reviewed, it is most likely a valid record and it is the key in error that needs to be deleted or replaced. This determination can be made only after properly investigating linkages to other relevant data files.

Finding and Fixing Missing Address Records using EpiInfo 6 Analysis

In the referential error: Address_id not in Address, a valid address_id for the patient may be substituted for the missing Address_id. Following is a step-by-step procedure for finding all of the valid Address_ids for a specific patient in the system. EpiInfo Analysis is bundled within STD*MIS and can be accessed through "Utilities, Access EpiInfo Programs, Analysis". The user may want to use the address_id with the closest As_of date, or, if this date is not available, the closest Add_dw to the date of the record with the missing Address_id. Do not make this decision lightly, as it will affect where morbidity will be assigned for any related case reports.

In Analysis type:
READ ADDRESS.DBF <ENTER>
SELECT PATIENT_ID = AXXXXXXXXXX@ <ENTER>
LIST ADDRESS_ID PATIENT_ID AS_OF STREET_1 CITY ADD_DW <ENTER>

To locate valid address_id's for additional patients, the user will need to first type in SELECT <ENTER> to clear the previous patient_id number. The user may then start with the next Patient_id at line 2 (select Patient_id = Axxxxxxxxxx@) for each additional address_id not in address problem. Repeat lines 2 - 4 as many times as needed. To save time and effort, the up arrow may be used to access the lines already written, making sure to substitute the new Patient_id after each Aselect@ statement clears out the old. To print the outcome of the statements after line 1, press the AF5" key before typing in the "select patient_id" statement. This will route the outcome to a printer. Pressing the AF5" key again will route the outcome back to the screen. Alternatively, the user may route the outcome to a text file on their hard drive by typing in:
ROUTE X:\XXXXX\XXXX.txt
Example: C:\stdreports\refint.txt.
Type <Route screen> after completing the searches to return the output to the screen.

Finding and Fixing Missing Address Records using FDBU

FDBU may also be used to locate valid address_id's for a specific patient. Load Address.dbf (F2), but do not browse it. Instead, chose F8 (Set). Use the down arrow key to highlight "Filter" and press <enter>. In the grey box, type in Patient_id='xxxxxxxxxx', then press <enter> twice. Next browse the database (F5). Only address records associated with the patient_id the user entered in the filter box will be displayed. If the address database is very large, it may take a few minutes to display all

the associated addresses. To remove the filter, press <escape> to close the browse, and choose F8 and “Filter” again. Delete the statement in the condition window to view the entire .dbf again, or replace the patient_id number with another one to view addresses associates with another patient. Be sure to end your session by deleting the last statement in the condition box.

SOLUTIONS

Note: Frb_id and Irb_id have two things in common: they both are 11 characters long (all other system-generated id numbers are 10 characters long) and both are derived by adding the 11th character to the end of another variable. Frb_id is the Fr_id plus the 11th character and Irb_id is the Ix_id plus the 11th character. The 11th character is always either the number A1" or A2”: “1” is used if it is a single infection; “2” is used in the case of a dual-infection. So if Frb_id is 01234567891, then Fr_id is 0123456789. Likewise, the reverse is true: if Ix_id is 0123456789, then Irb_id is either 01234567891 or 01234567892. Always start with A1" first since it represents the first infection in the situation of a co-infection with either the interview or field record. This shortcut may save time in trying to find linked records, allowing the user to skip a step in the process of data cleaning.

AKA

This file contains alternate names that patients may be known as.

Problem: Patient_id not in Patient.dbf

The user may wish to browse the Patient.dbf to make sure that the record has not been marked for deletion. Since Patient_id is the only linking variable in this DBF, if it is not located in patient.dbf the record is usually not salvageable. The resolution is to mark the AKA record for deletion.

CASE

All patient records, named or not, that have morbidity, field records, or interview records have a related case file or files. If there is more than one system id other than Case_no within a record (i.e. morb_id, irb_id or frb_id), be sure to check all of the potential related files before deciding to mark a Case record for deletion. Although the link that has the error may need to be deleted itself, the case record may still be a valid record because of its link to other files.

Problem: Morb_id not in Morbrept.dbf

The case record has 2 other variables that, if present, can possibly be used to find the solution to this problem. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well. If the linked Morbidity records are marked for deletion, then the Morb_id in case may need be removed if there are valid field or interview records associated with the case record. Users should be aware that adding or deleting in Morbrept affects the case counts that are reported to CDC and used in morbidity reports. With this in mind, the user should be aware of local program policies regarding morbidity and have the authority to make such changes. If the user does not have such authority, it is strongly recommended that they inform proper parties of the effects of such proposed changes to morbidity before effecting any changes to morbidity files.

Frb_id - Since this variable is the same as Fr_id (described above), use the Fr_id in Fr.dbf to find the Event_id. Go to Morbrept.dbf and search for the Fr.event_id. If the records are properly linked then this should find the Morbrept record that belongs to the Case record. The user can decide which record to change, Morbrept or case, to match the other. An example of this solution is listed above in the Documentation section.

Irb_id - The same method described above for Frb_id can be used using Irb_id and Ix_id to resolve this problem. An alternative method, in case the event_id can not be found in Morb_rept is to use the Ix_id (since it is the same as the Irb_id, as described above) to find the Patient_id. The Patient_id can be used to search Morbrept.dbf. Keep in mind that a patient may have more than one morbidity records, but comparison with the dates associated with the two records, Case and Morbrept, usually points to which record is the best choice for a match. The user can decide which record to change, Morbrept or Case, to match the other. Example:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Morb_id xxxxxxxxxx not in Irb	ix_id	>Ira.pat_id	>Irb.pat_id*	evaluation	choose	Replace Irbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	various recs	closest rec.	

> - to file.field name xxxxxxxxxx - variable value *may be multiple records

Problem: Irb_id not in Irb.dbf

The case record has 2 other variables that, if present, can possibly be used to find the solution to this problem. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well. If all linked Ir records are marked for deletion, then the Irb_id in case should be removed.

Morb_id - Locate the morb_id of the Case record in Morbrept.dbf. From Morbrept record get the Event_id. Go to Ira.dbf and search for the Morbrept.event_id. From the Ira record get the Ix_id. Go to Irb.dbf and search for the Ira.ix_id. From Irb get the correct Irb_id and place it in the Case record. Example:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Irb_id xxxxxxxxxx not in Irb	morb_id	>Morb.eventid	>Ira.Ix_id	>Irb.Ix_id	>Irb.irb_id	Replace Irbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	

> - to file.field name xxxxxxxxxx - variable value

Morb_id is also found in Std126 (congenital cases).

Frb_id - From Frb_id determine Fr_id (Frb_id minus the 11th character.) Find the Fr_id of the Case record in Fr.dbf. From the Fr.dbf record get the Event_id. Go to Ira.dbf and search for the Fr.event_id. From the Ira record get the Ix_id. Go to Irb.dbf and search for the Ira.ix_id.

From Irb get the correct Irb_id and place it in the Case record. Example:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Irb_id xxxxxxxxxx not in Irb	Fr(b)_id	>Fr.event_id	>Ira.Ix_id	>Irb.Ix_id	>Irb.irb_id	Replace Irbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	

> - to file.field name xxxxxxxxxx - variable value

Problem: Frb_id not in Frb.dbf

The case record has 3 other variables that, if present, can possibly be used to find the solution to this problem. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well. If all other linked Fr records are marked for deletion, then the Frb_id in case should be removed.

Case_no - Go to Referral.dbf and search for the Case_no from the Case record with a Ref_basis = AT1" or AT2.@ The Frb_id is also in this file. Place the Frb_id in the Case record. Example:

Expression case_no = ›xxxxxxxxxx= .and. ref_basis=-T1'

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Frb_id xxxxxxxxxx not in Frb	Case_no	>ref.basis = T1	>ref.Frb_id			Replace Frbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx		xxxxxxxxxxx			

> - to file.field name xxxxxxxxxx - variable value

Morb_id - Go to Morbrept.dbf and search for the Morb_id of the Case record. From Morbrept record get the Event_id. Go to Fr.dbf and search for the Morbrept.event_id. Go to Referral.dbf and search for the Fr.Fr_id. From Referral, get the correct the Frb_id and place it in the Case record. Alternatively, if the Fr_id is not located in Referral.dbf, the user can create an Frb_id by adding a one (1) or a two (2), in the case of a dual infection on the field record, to the end of the Fr_id. Search Frb.dbf for the created Frb_id(s) to verify its existence. Place the Frb_id in the Case record. Example:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Frb_id xxxxxxxxxx not in Frb	morb_id	>Morb.eventid	>Fr.fr_id	>Frb.frb_id		Replace Frbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

Irb_id - Since this variable is the same as Ix_id (described above), use the Ix_id in Ira.dbf to find the Event_id. Go to Frb.dbf and search for the Ira.event_id. Example:

Errors	**Dbf: Case**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Frb_id xxxxxxxxxxx not in Frb	Ix (irb)_id	>Ira.event_id	>Frb.frb_id			Replace Frbid w/ xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxxx			

> - to file.field name xxxxxxxxxx - variable value

FR

Since these problems deal with Field Records, a hard copy, if available, of the corresponding record may help in the resolution of these types of errors. At the very least, hard copies can provide a check on certain validations, such as name of patient, referral basis, and patient home address. Users may want to double check these records before implementing solutions.

Problem: Fr_id not in Referral.dbf

The FR record has 2 other variables (counting Frb_id though not in this DBF) that, if present, can possibly be used to find the solution to this problem. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well.

Frb_id - Since Frb_id is Fr_id without the last character, then placing an additional character onto Fr_id, numbers A1" or A2" (always start with A1" first), will give the Frb_id. Using this created Frb_id, conduct a search in Referral. If this does not lead to a Referral record, the data manager may want to try to search on the created Frb_id in Case.dbf. If a record is found in Case, use the case number to perform a search in Referral. Since the Case_no can be repeated in Referral, the user will need to make a determination between the records found. Do not assume that a found record is the correct choice without doing some research. An example of the second procedure follows:

Errors	**Dbf: Fr**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Fr_id xxxxxxxxxx not in Ref	Frb_id	>Case.case_no	>Ref.fr_id	2nd Ref.fr_id	3rd Ref.fr_id	Replace Fr_id w/ xxxxxxxxxx in Fr
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxx x	xxxxxxxxxx	

> - to file.field name xxxxxxxxxx - variable value

Event_id - Go to Morbrept and search for the Event_id of the Fr record. If found, search the Morb_id in Case. Use the found Case_no to search Referral. Since the Case_no can be repeated in Referral, the user will need to make a determination between the records found. Do not assume that a found record is the correct choice without doing some research. This method will not work with partner and cluster records. A compound search statement may help with this issue.

Expression case_no = ›xxxxxxxxxx‹ .and. ref_basis=‹T1'

Errors	**Dbf: Fr Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Fr_id xxxxxxxxxx not in Ref	Event_id	>morb.morbid	>Case.caseno	>Ref.fr_id	2nd ref.fr_id	Replace Fr_id w/ xxxxxxxxxx in Fr
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	

> - to file.field name xxxxxxxxxx - variable value

Problem: Patient_id not in Patient.dbf

The Fr record has 3 other variables that, if present, can possibly be used to find the solution to this problem. Since Patient_id is stored in many databases, the user should examine all linked files to determine if a different Patient_id is used anywhere else in the system. The Event_id, **Key Linkages** document, and the **Fields List** document will inform the user of all of the related databases that may be searched. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the patient record in Patient.dbf, if that information is known.

Frb_id or Fr_id - This method will not work for partner or cluster records. Search for the Frb_id or Fr_id in Referral. If there is a Case_no associated with the Referral record, search Case on the Case_no. Use either the Morb_id to search Morbrept or the Irb_id (if present) to search Irb. Each of these databases also has a Patient_id to compare with the Patient_id in Fr.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database found using either the **Fields List** document or the **Key Linkages** document to search for the Event_id of the specific record. Check only those databases that also have Patient_id stored as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

Problem: Address_id not in Address

The Fr record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be best resolved by using EpiInfo 6 Analysis to read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient‹s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

Problem: Short_id not in Shortfr.dbf

Short_id is only found in three database files: Fr, Shortfr, and Stdsys. The Stdsys file will not aide in the resolution of this type of error. In fact, if searching Shortfr.dbf does not yield that the

record has been marked for deletion, there are only two options: delete this record in Fr or recreate the record in Shortfr. If the hard copy of the interview record on which this record is related is available, the preferable option is to recreate the record.

FRA

Since these problems deal with Field Records, a hard copy, if available, of the corresponding record may help in the resolution of these types of errors. At the very least, hard copies can provide a check on certain validations, such as name of patient, referral basis, and patient home address. Users may want to double check these records before implementing solutions.

Problem: Fr_id not in Fr.dbf

The user may want to first confirm that this is a valid record. If the hard copy of the field record is available, this may help resolve this error. The user will want to check Frb to see if the record has been marked for deletion. Go to Referral.dbf as well to search for the Frb_id. If the Fr_id is the same as the Fr_id in Fra.dbf, then there should have been a corresponding Error found by the program in this file. If they are different, then the user may want to search for the new Fr_id in Fra, to see if there was a similar Fra record created with a different Fr_id at another point in time. If this is so, then the user may want to delete the record that was in error. If the record is valid, the hard copy of the Fr will make it much easier to recreate the record. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well.

FRB

Since these problems deal with Field Records, a hard copy, if available, of the corresponding record may help in the resolution of these types of errors. At the very least, hard copies can provide a check on certain validations, such as name of patient, referral basis, and patient home address. Users may want to double check these records before implementing solutions.

Problem: Frb_id not in Referral.dbf

Since Frb_id is essentially the same as Fr_id, there are two variables that can be used to help resolve this type of error. If all other linked records are marked for deletion, then this record may need to be marked for deletion as well.

Fr_id - Search for Fr_id in Referral.dbf. Place the correct Frb_id in the Referral record. Another possible method to resolve this problem using Fr_id: if the Fr was created from a reactor, use the Event_id from Fr and search for the Event_id in Reactor. Example:

Errors	**Dbf: Fr Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Frb_id xxxxxxxxxx not in Ref	Fr(b)_id	>Fr.event_id	>Reac.reacid			Replace Frb_id w/ xxxxxxxxxx in Frb
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx			

> - to file.field name xxxxxxxxxx - variable value

Frb_id - Go to case and search for Frb_id. Use the case_no to search in Referral. Keep in mind that Case_no can be repeated several times in Referral. Compare Ref_dw and Ref_tw to

Frb_tw and Frb_tw to decide which is best for record for use.

IRA

Since these problems deal with Interview Records, a hard copy, if available, of the corresponding record may assist in the resolution of these types of errors. At the very least, hard copies can provide a check on certain validations, such as name of patient and patient home address. Users may want to double check these records before implementing solutions.

Problem: Ix_id not in Irb.dbf

The Ira record has 2 other variables that, if present, can possibly be used to find the solution to this problem. However, a hard copy of the interview record may be helpful if it is determined that a record needs to be recreated. Within the Patient Registry, under Interview Record, perform a search on the Control Number of the interview record. This should yield all of the information needed to locate the hard copy of the record. If no other linked records are found, the Ira record may need to be marked for deletion.

Event_id - Go to Morbrept and search for the Event_id. Using the Morb_id, search in Case. If an Irb_id is found, search Irb.dbf for it. Example:

Errors	**Dbf: IRA**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Ix_id xxxxxxxxxx not in Irb	event_id	>Morb.morb_id	>Case.irb_id	>Irb.ix_id		Replace Ix_id w/ xxxxxxxxxx in Ira
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

Irb_id - Since Irb_id is Ix_id with the addition of an extra character on the end, add either a A1" or A2" (always start with A1" first) to create the Irb_id and search Irb for the Irb_id. If found, compare the Ix_id with the missing Ix_id. If The Irb record is marked for deletion, it may need to be undeleted, or the Ira record may need to be marked for deletion as well.

Problem: Patient_id not in Patient.dbf

The Ira record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Since Patient_id is stored in many databases, the user should examine all linked files to determine if a different Patient_id is used anywhere else in the system. The Event_id, **Key Linkages** document, and the **Fields List** document will inform the user of all of the related databases that may be searched. If all other linked records are marked for deletion, then this record may need to be deleted as well. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the patient record in Patient.dbf.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database found using either the Field List document or the **Key Linkages** document to search for the event_id of the specific record. Check only those databases that also have Patient_id stored

as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in Patient.

Ix_id - The user may want to search Irb.dbf with the Ix_id to make sure that both have the same Patient_id. If they are the same, then Irb should have this same error (Patient_id not in Patient). Search Case.dbf on the Irb_id. If there is a Morb_id in Case, use it to search Morbrept. Compare Event_ids between Ira and Morbrept, as well as Patient_ids. If there is a difference in Patient_ids between the two, document the Morbrept.patient_id. Check the new patient_id in Patient.dbf to insure that it is a valid record as well.

Problem: Address_id not in Address.dbf

The Ira record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant address_id. Then a listing should be made of all of that patient-s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad address_id. The search can also be done in FDBU, setting the filter as described on page 6.

IRB

Since these problems deal with Interview Records, a hard copy, if available, of the corresponding record may help in the resolution of these types of errors. At the very least, hard copies can provide a check on certain validations, such as name of patient and patient home address. Users may want to double check these records before implementing solutions. Also, if Ix_id is not in Ira.dbf and Irb_id is not in Case.dbf, the record with these problems probably should be marked for deletion, since all of its linking records are missing.

Problem: Ix_id not in Ira.dbf

The Irb record has 1 variable that can possibly be used to find the solution to this problem. Go to Case.dbf and search for Irb_id. If present, use Morb_id to find the corresponding Morbrept.dbf record. Search Ira for the Morbrept.event_id. If Morb_id is not present but Frb_id is, use Fr_id (Frb_id minus the last character), to search Fr.dbf.

Problem: Irb_id not in Case.dbf

The Irb record has 1 variable that can possibly be used to find the solution to this problem. Go to Ira.dbf and search for Ix_id. Use Ira.event_id to search Fr or Morbrept. If found in Morbrept, search case for the Morb_id; Irb_id may need to be placed in the found record. If found in Fr, add the number A1" (or A2" as applicable) to the end of Fr_id to make Frb_id. Use Frb_id to search Case. If found, Irb_id may need to be placed in the record. Example using Morb_id:

Errors	**Dbf: Fr Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Irb_id xxxxxxxxxx not in Case	Ix_id	>Ira.event_id	>Morb.morbid	>Case.caseno		Place Irb_id xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

MORBREPT

In dealing with these types of records it is important to understand the full implications of deleting or adding records to this file. The name of the file is exactly what it implies, the morbidity report file. Case.dbf actually stores the types of disease and diagnosis being reported, but Morbrept.dbf is the file that makes each case a report. Adding or undeleting records actually adds to the number of cases that are reported to CDC and are shown in disease reports. Marking records for deletion reduces the number of cases reported. The user should be familiar with local program practices in regard to this issue. If the user does not have authority to affect case counts, the user should consult program management before implementing any changes. This does not mean that the user should not pursue the solution to problems, just that the user should be aware of the effects of changes and communicates the changes to proper personnel, if needed, before implementing the changes. Once again it should be pointed out that resolutions of referential errors should first be implemented on a copy of the database and not the production, or 'live', version. Once the user is satisfied with the solutions they should be implemented on the live data only after the data is backed up.

Problem: Morb_id not in Case.dbf

The Morbrept record has 1 variable that can possibly be used to find the solution to this problem.

Event_id - Use Event_id to search either Ira.dbf or Fr.dbf. If Ira_id is found, create Irb_id from Ix_id, described earlier, or search Irb.dbf with Ix_id to find Irb_id. Go to Case.dbf and search for the Irb_id. Morb_id may need to be placed within the record found. If Fr_id is found, add the number A1" (or A2", as needed) to create Frb_id. Search Case.dbf for the Frb_Id. Morb_id may need to be placed in this record. If either of these solutions does not work, Event_id should also be used to search other files such as Test.dbf to see if morbidity is linked. If there are no other links to this file within the system, it is probable that this record needs to be marked for deletion. Example using Event_id to FR.dbf:

Errors	**Dbf: Fr Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Morb_id xxxxxxxxxx not in Case	Event_id	>Fr.fr_id	>Fr.fr(b)_id	>Case.case_no		Place Morb_id xxxxxxxxxx in Case
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxxx	xxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

Problem: Patient_id not in Patient.dbf
The Morbrept record has 2 other variables that, if present, can possibly be used to find the solution to this problem.

Morb_id - Go to Case and search for the Morb_id. If present, use Frb_id minus the last character to form Fr_id and search Fr.dbf; compare Patient_ids if a record is found. Else, if present, use Irb_id minus the last character to form Ix_id and search Ira.dbf; compare Patient_ids if a record is found. If the Patient_id that is found under either of these situations is the same as the Patient_id in Morbrept, then these files will also have the same referential integrity error. The patient record may need to be recreated in Patient.dbf. If the Patient_ids found are dissimilar, the user may want to make sure that the new Patient_id is a valid patient record before placing it in the Morbidity record with the error.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. Possible databases links may be found in Test, Fr, and Ira. The user must choose a linked database found using either the **Fields List** document or the **Key Linkages** document to search for the event_id of the specific record. Check only those databases that also have Patient_id stored as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

Problem: Address_id not in Address.dbf
The Morbrept record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient-s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

OOJLOG
The Out Of Jurisdiction LOG contains information on the tracking of events that have originated from or have been sent to areas outside of the local jurisdiction.

Problem: Patient_id not in Patient.dbf
The Oojlog record has 1 variable that can possibly be used to find the solution to this problem.

Event_id - Using the **Key Linkages** and/or **Fields List** documents, find the other databases this record may be potentially linked with. Fr and Test databases are probably the most common links to Oojlog. Check the linked records on Event_id to see if they have a different Patient_id associated with them. Most likely, however, they will have the same Patient_id and the same referential integrity errors will be listed, if checked. If this record is valid with other

linked events, then recreation of the Patient record may be the best solution to this problem. The User may need to contact the initiating agency, located in the errant record, to find out the particulars of the individual patient to accurately recreate the Patient record.

Problem: Address_id not in Address.dbf

The Oojlog record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient-s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

P_LINK

This database deals with the cross referencing or AJump@ option within STD*MIS. Unlike most of the other files, this database only relates to one other DBF, Patient.dbf. The variable Rel_id is the Patient_id of the related patient record. These relationships are paired so each relation actually has two lines of data in the data file. For example, Patient #1 is cross-referenced with patient #2; then on one line of data, #1 has the Patient_id and #2 has the Rel_id. On the second line of data, Patient #2 has the Patient_id and #1 has the Rel_id. This creates the link between the two patient records. However, since this DBF relates only to patient, there are few options to resolve these types of referential errors.

Problem: Patient_id not in Patient.dbf

If the individual record does not have both errors (Patient_id not in Patient and Rel_id not in patient), then the user can search Patient.dbf for the Rel_id, using the variable name APatient_id.@ Once this person is found, some research may be in order to find the corresponding name of the linked record. This may mean checking with field or surveillance staff, or seeing if the related name is in the AKA file. Once the name is found, search STD*MIS for the system id to see if the patient record still exists. If it does, document the correct Patient_id and use it to replace the problem id in P-link.

Problem: Rel_id not in Patient.dbf

If the individual record does not have both errors (Patient_id not in Patient and Rel_id not in patient), then the user can search Patient for the Patient_id. Once this person is found, some research may be in order to find the corresponding name of the linked record. This may mean with field or surveillance staff, or seeing if the related name is in the AKA file. Once the name is found, search STD*MIS for the system id to see if the patient record still exists. If it does, document the correct Patient_id and use it to replace the problem id (Rel_id) in P-link.

NOTE: If the record is has the problem that both Patient_id and Rel_id are missing from patient, then there is no way to recreate the link. The record, and its corresponding double, should be

marked for deletion.

REACTOR

This database contains the information gathered for surveillance tracking.

Problem: Patient_id not in Patient.dbf

The Reactor record has 2 other variables that, if present, can possibly be used to find the solution to this problem.

Reactor_id - Go to Test.dbf and search for the Reactor_id. If found, compare the Patient_ids. Another method may be to go to Referral.dbf and search for the Reactor_id. Using the Fr_id in the record found in Referral, search Fr.dbf. Compare Patient_ids between Fr and Reactor. If the Patient_ids are different in either scenario, use the newly found Patient_id. The user may want to verify that the found Patient_id is in the Patient file.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database, found using either the **Fields List** document or the **Key Linkages** document, to search for the Event_id of the specific record. Check only those databases that also have Patient_id stored as a field. Possible databases links may be found in Test.dbf and Fr.dbf. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

Problem: Reactor_id not in Test.dbf

The Reactor record has 1 other variable that, if present, can possibly be used to find the solution to this problem.

Event_id - Both Test and Reactor files have Event_id and Receive_dt as variables; an option to trying to find the missing record may be to search on both variables at the same time in Test. The locate statement would be as follows:

Expression event_id = ›xxxxxxxxxx‹ .and. receive_dt = ctod(›xx/xx/xx‹)

Errors	**Dbf: Reactor Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Reactor_id xxxxxxxxxx not in Test	Receive_dt	>Test - search				Place Reac_id xxxxxxxxxx in Test
	xx/xx/xxxx	Reac_id & Rec_dt				

> - to file.field name xxxxxxxxxx - variable value

REFERRAL

This database links field records, test results, OOJ, and case data to one another. The referral basis is also stored within this file. Since this database touches so many different files, the user should be careful in deciding whether to mark a record for deletion or delete/correct a specific variable as needed. The user should check all related records (Case, Fr, Frb, Oojlog, and Reactor) before marking any of these for deletion. Also, there may be second generation linkages

that, once a record in this file is deleted, may cause referential integrity problems in other databases. For example, Reactor_id is also in Test and Survclos.

Problem: Fr_id not in Fr.dbf

The Referral record has 1 other variable that can possibly be used to find the solution to this problem. Refer to the **Fields List** and **Key Linkages** documents to see what databases are linked, but are not a part of the Referral file. If all linked Fr records (Fra, Frb, and/or Shortfr) and other linked records are marked for deletion, then this record also needs to be marked for deletion.

Event_id - Search on the variable id in any of the other databases (Case, Oojlog, and Reactor) that may be present to find an Event_id. Use that Event_id to search Fr. If found compare Fr_ids. If not found, check all related files as stated above. Example using Case:

Errors	**Dbf: Referral Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Fr_id xxxxxxxxxx not in Fr.dbf	case_no	>Case.morbid	>Morb.eventid	>Fr.fr_id		Place Reac_id xxxxxxxxxx in Test
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

Problem: Frb_id not in Frb.dbf

The Referral record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Refer to the **Fields List** and **Key Linkages** documents to see what databases are linked, but are not a part of the Referral file. If all linked Fr records (Fr and Fra) are deleted, then this variable may need to be deleted in this file. If linked Fr records are valid, then the Frb record may need to be recreated. If all linked Fr records and other records linked to this particular record are marked for deletion, then this record also needs to be marked for deletion.

Fr_id - Check this id to make sure it matches Frb_id, minus the last character. Though unlikely, in the event that they do not match, go to Frb and search on the non-matching Fr_id, adding the number A1" or A2" to create an Frb_id. If a record is found in Frb, then the Frb_id in Referral needs to be changed to match the valid Fr_id.

Case_no – ***NOTE:*** *This solution is only valid for reactors (Ref_basis = T1 or T2). Partner and cluster records (ref_basis = P1, P2, P3, S1, S2, S3, A1, A2, A3) have their Original Patient's (OP) case_no in their referral record, so it can not be used to find valid Frb_ids.* Go to Case and search for the Case_no of the errant record. If present, and if the frb_id in the found Case record is different that the missing frb_id, search Frb with case.frb_id to verify it is a valid id and record. Use the new Frb_id in Referral. If the Case.frb_id is the same as the problem record in Referral, check the corresponding records in the FR files (FR, FRA, and FRB) to see if the components of the FR have been marked for deletion. If the components have been marked

for deletion, then it is likely the field record was meant to be deleted from the system and has probably been re-entered with different id numbers. If this is the case (verified by checking to see if there is a duplicate reactor on the patient with a different fr_no), then all components of this reactor need to be marked for deletion.

Another way to use Case_no to resolve this error would be to utilize the procedures listed directly above using the case_no to find event_id in Morbrept or IRA. See diagram preceding this topic.

Problem: Case_no not in Case.dbf

NOTE: *This solution is only valid for reactors (Ref_basis = T1 or T2). Partner and cluster records (ref_basis = P1, P2, P3, S1, S2, S3, A1, A2, A3) have their Original Patient's (OP) case_no in their referral record.*

The Referral record has 4 other variables that, if present, can possibly be used to find the solution to this problem. Refer to the **Fields List** and **Key Linkages** documents to see what databases are linked but are not a part of the Referral file. If no corresponding Case record is found, or if the case record and its related records are marked for deletion, then this variable may need to be removed from the Referral record.

Fr_id - Go to Fr and search for the Fr_id of the errant record. Use Fr.event_id to search either Morbrept or Ira. If using Morbrept, search Case.dbf with Morb.morb_id. If using Ira, create Irb_id by adding either @1" or A2" to the end of Ira.ix_id. Then search Case with the new Irb_id. The options of going through Morbrept or Ira may lead to two separate Case records. The user can choose which record is best suited to replace the Case_no in Referral. Example using Ira:

Errors	**Dbf: Referral Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Case_no xxxxxxxxxx not in Case	Fr_id	>Fr.event_id	>Ira.ix_id	Irb_id	>case.case_no	Place Caseno xxxxxxxxxx in Referral
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	

> - to file.field name xxxxxxxxxx - variable value

Frb_id - Use Frb_id to search Case. If a valid record is found, substitute the new Case_no for the old in Referral.

Oojlog - Go to Oojlog and search for the Ooj_id of the errant record. Use Ooj.event_id to search either Morbrept or Ira. If using Morbrept, search Case.dbf with Morb.morb_id. If using Ira, create Irb_id by adding either @1" or A2" to the end of Ira.ix_id. Then search Case with the new Irb_id. The options of going through Morbrept or Ira may lead to two separate Case records. The user can choose which record is best suited to replace the Case_no in Referral.

Reactor_id - Go to Reactor and search on the Reactor_id of the errant record. Use Reactor.event_id to search either Morbrept or Ira. If using Morbrept, search Case.dbf with Morb.morb_id. If using Ira, create Irb_id by adding either @1" or A2" to the end of Ira.ix_id, then

search Case with the new Irb_id. The options of going through Morbrept or Ira may lead to two separate Case records. The user can choose which record is best suited to replace the Case_no in Referral.

Problem: Reactor_id not in Test.dbf

The Referral record has 4 other variables, if present, that can possibly be used to find the solution to this problem. Most of these options involve using the Event_id to find a corresponding record in Test.dbf, with the exception of using the Case_no. If no corresponding Test record can be found, then this variable may need to be deleted from the Referral record.

Event_id - Search on the variable id in any of the four other databases that may be present to find an Event_id. Use that Event_id to search Test. If found, use the Event_id and Receive_dt to search Test. Example:

Expression event_id = ›xxxxxxxxxx‹ .and. receive_dt = ctod(›xx/xx/xx‹)

Errors	**Dbf: Reactor**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Reactor_id xxxxxxxxxx not in Test	Receive_dt	>Test - search				Place Reacid xxxxxxxxxx in Test
	xx/xx/xxxx	Reac_id & Rec_dt				

> - to file.field name xxxxxxxxxx - variable value

Case_no - Go to Case and search for the Case_no of the errant record. Once found, the Morb_id, Irb_id, or Frb_id can be used to find an Event_id. Morbrept can be searched directly using the Morb_id. Using either Frb_id or Irb_id will require creating Fr_id or Ix_id, respectively, by removing the last character from each variable. Use Fr_id to search Fr.dbf and Ix_id to search Ira.dbf. Use the found Event_id to search Test.dbf.

Problem: Ooj_id not in Oojlog.dbf

The Referral record has 4 other variables that, if present, can possibly be used to find the solution to this problem. Most of these options involve using the Event_id to find a corresponding record in Oojlog.dbf, with the exception of using the Case_no. If unable to find a corresponding Oojlog record, this variable may need to be deleted from the Referral record.

Event_id - Search on the variable id in any of the four other databases that may be present to find an Event_id. Use that Event_id to search Oojlog.

Case_no - Go to Case and search for the Case_no of the errant record. Once found, the Morb_id, Irb_id, or Frb_id can be used to find an Event_id. Morbrept can be searched directly using the Morb_id. Using either Frb_id or Irb_id will require creating Fr_id or Ix_id, respectively, by removing the last character from each variable. Use Fr_id to search Fr.dbf and Ix_id to search Ira.dbf. Use the found Event_id to search Oojlog.dbf.

REMOTE
This database gives an snapshot of uploaded records in the local system.

Problem: Patient_id not in Patient.dbf
Since this database deals with uploaded records, this is one of the more difficult types of fixes to effect and may need some interaction with remote areas to garner data needed to properly fix the error(s). Also, unlike other uploaded files, patient records become a part of the local STD*MIS (i.e. they become native records) while other uploaded events, such as Test, FRs, etc. are stored in their own databases, separate from the local STD*MIS files. (For a more complete description of Uploaded Records, see the **STD*MIS Uploaded Records** document.) The cause of this problem may be rooted in an uploaded patient record that did not get written into the local system for some reason, but its related files *were* written to the local system. Or a patient record may have been deleted from the system and the remote record did not get deleted. **Note:** These fixes are unique to Remote.dbf and should not be used with any other files.

Remote systems - The Site variable in Remote tell the user from which remote area the record originated. By contacting the remote area and having them perform a record search in their STD*MIS on the Patient_id, the user can find the name of the patient and other pertinent information to help resolve this problem. With the correct patient information, perform a record search with the local STD*MIS. If found, the user can replace the Patient_id in Remote with the correct Patient_id from the found record. If not found with the system, the user can recreate the patient record using information gained from the remote area. See the ARecreating Selected Records@ section on Page 26 for instructions on recreating patient records.

Upldlink - If the Patient_id is not found in the remote area, there is another option to examine. The Patient_id in Remote is the Local_id in Upldlink.dbf. The Remote_id in Upldlink is the remote site=s Patient_id. The user can search Upldlink using Local_id (Patient_id) and find the corresponding Remote_id (the remote area=s Patient_id). Then the user can then ask the remote area to conduct a record search.

Other uploaded records - If the above two methods fail to yield a satisfactory answer to the referential integrity error, there is a third method to attempt to rectify the problem. Using Rec_type and Rec_id in Remote, the user can explore the uploaded record databases for a potentially valid Patient_id. This can be used to conduct a record search at the remote site or may be found in the Upldlink.dbf. Any found Patient_ids found in the uploaded records are Patient_ids from remote areas, not the local STD*MIS. Below is a list of Remote Rec_types, what their Rec_ids correspond to in STD*MIS, and their corresponding upload database:

Rec_type	**Type file**	**Rec_id**	**Database**
CS	Congenital Syphilis	Cs_id	Csupld.dbf
FR	Field Records	Fr_id	Frupld.dbf
IR	Interview Records	Ix_id	Irupld.dbf

TE	Lab Tests	Test_id	Testupld.dbf
MO	Morbidity Files	Morb_id	Morbupld.dbf
VI	Visits to Clinic	Visit_id	Visupld.dbf

Using the table above, the user can search the appropriate upload database for the record matching record to the Remote record. As Patient_id is a variable in each of these files, the user has the remote area Patient_id to utilize in helping to resolve the error. The user should repeat the steps of Remote area and Upldlink above. Since the remote record is a reflection of the uploaded database record, if the database record is marked for deletion, then the remote record can be marked for deletion as well.

RX

This file contains treatment records.

Problem: Patient_id not in Patient.dbf

The Rx record has 1 other variable that can possibly be used to find the solution to this problem.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database found using either the **Fields List** or the **Key Linkages** document to search for the event_id of the specific record. Possible databases links may be found in Test, Fr, and Ira. Check only those databases that also have Patient_id stored as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

STD126

This database contains information pertaining to Congenital Syphilis cases and reporting.

Problem: Patient_id not in Patient.dbf

The Std126 record has 1 other variable that can possibly be used to find the solution to this problem.

Event_id - use the Event_id to search either Fr, Reactor, or Test to see if there are any corresponding records in either of these files. If a record is found, compare the Patient_ids. If these Patient_ids are the same, then the patient record may need to be recreated on this infant. If there is a valid Mother_id (Patient_id of the mother), there may be additional information available to help complete this record.

Problem: Mother_id not in Patient.dbf

The Std126 record has 1 other variable that can possibly be used to find the solution to this problem. This variable, Mother_id, is the Patient_id of the mother. When comparing other databases to STD126, Mother_id should be compared with patient_id.

Mom_add_id - This variable is the mother's Address_id, which may or may not be the same as infant-s address. Search the Address.dbf for the Address_id (Mom_add_id); if a matching record is found, make sure it is not the infant-s record by comparing Patient_id in

Std126 and Address.dbf. Assuming that the Patient_id from Std126 is a valid id and the Address.patient_id is different than the infant=s Patient_id, this is most likely the Mother_id. It is easy enough to check this by searching for the found Patient_id in Patient.dbf. If no valid Patient_id is found then the patient record may need to be recreated with the new Patient_id also used as the Mother_id.

Problem: Address_id not in Address.dbf

The Std126 record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient=s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

Problem: Mom_add_id not in Address.dbf

The name of this variable is misleading and it will only be found in this database. For all purposes this variable is the same as Address_id. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem, though here Mother_id is the variable used since the Patient_id of this record belongs to the infant child. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id (Mother_id) - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id (Mother_id) of the record with the errant Address_id. Then a listing should be made of all of that patient=s Address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad address_id. The search can also be done in FDBU, setting the filter as described on page 6.

Problem: Morb_id not in Morbrept

The Std126 record has 1 other variable that can possibly be used to find the solution to this problem.

Event_id - Go to Morbrept and search for the Event_id of the errant record. If not found, this Morbrept record may need to be recreated. If found, but marked for deletion, the search may need to be continued. The user may want to check the Event_id in Test and Fr to make sure that the Std126 record is a valid record.

SURVCLOS

This database contains information regarding dispositions of cases assigned to surveillance follow-up.

Problem: Reactor_id not in Reactor.dbf

The Survclos record has 2 other variables, if present, that can possibly be used to find the solution to this problem. Using the **Fields List** document, the user may want to verify that this is a valid record. If all links are marked for deletion, then this record will also need to be marked for deletion.

Reactor_id - Go to Referral and search for the Reactor_id. If the record has either a Fr_id or an Ooj_id, search their respective data bases for a corresponding record. Use the Event_id from the record to search Reactor. If the Referral record has a Case_no, use it to search Case. Search Morbrept with the Case.morb_id. Use the Morb.event_id to search Reactor. Example using Case_no:

Errors	**Dbf: Survclos Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Reactor_id xxxxxxxxxx not in React	>Ref.case_no	>Case.morb_id	>Morb.eventid	>React.reacid		Place Reacid xxxxxxxxxx in Survclos
	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx		

> - to file.field name xxxxxxxxxx - variable value

Reactor_id can also be used to search Test.dbf, which may yield adifferent Event_id. Use Test.event_id to search Reactor.

Morb_id - Go to Morbrept and search for the Survclos.morb_id. Use the Morb.event_id to search Reactor.dbf.

Problem: Morb_id not in Morbrept.dbf

The Survclos record has 1 other variable that can possibly be used to find the solution to this problem.

Reactor_id - Go to Referral and search for the Reactor_id. If the record has either a Fr_id or an Ooj_id, search their respective data bases for a corresponding record. Use the Event_id from the record to search Morbrept. If the Referral record has a Case_no, use it to search Case. Search Morbrept with the Case.morb_id . Example using Case_no:

Errors	**Dbf: Survclos Date: XX/XX/XXXX Completed by: H. D. Clown**					Solutions
Morb_id xxxxxxxxxx not in Morbrept	>Ref.case_no	>Case.morb_id				Place Morb_id xxxxxxxxxx in Morbrept
	xxxxxxxxxx	xxxxxxxxxx				

> - to file.field name xxxxxxxxxx - variable value

Reactor_id can also be used to search Test.dbf, which may yield another Event_id. Use Test.event_id to search Morbrept.

TEST

This database contains test result information

Problem: Patient_id not in Patient.dbf

The Test record has 2 other variables that, if present, can possibly be used to find the solution to this problem. If through checking all other linked records, no valid Patient_id is found, the user may choose to mark this record for deletion, keeping in mind there may be other linked records to this file, or choose to recreate the patient record.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database found using either the **Fields List** document or the **Key Linkages** document to search for the Event_id of the specific record. Databases with possible links: Fr, Morbrept, Ira, and Reactor. Check only those databases that also have Patient_id stored as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

Reactor_id - Go to Reactor and search for the Reactor_id of the errant record. If the Patient_id found here matches the one in Test, use the Reactor_id to search Referral. From Referral there are a couple of other databases that can be checked for Patient_id comparison.

Problem: Address_id not in Address.dbf

The Test record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient-s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

Problem: Reactor_id not in Reactor.dbf

The Test record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Using the **Key Linkages** document, the user may want to verify that this field should have data in it for this particular record. If all links are marked for deletion, then this field should be deleted from the Test record.

Event_id - Go to Reactor and search for Test.event_id. If found compare the information and make determination whether to make use of it or not. If Test.event_id not found in Reactor, use this Event_id to search Fr, Morbrept, or Oojlog. If any of these are found, use their corresponding key ids, Fr_id, Morb_id, or Ooj_id respectively, to search Referral. If a record is found in Referral, compare the Reactor_id if one is present. Example:

Errors	**Dbf: Test**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Reactor_id	Event_id	>Fr.Fr_id	>Ref.reactor_id			Place Reacid
xxxxxxxxxx						xxxxxxxxxx

not in React	xxxxxxxxxx	xxxxxxxxxx	xxxxxxxxxx			in Test

> - to file.field name xxxxxxxxxx - variable value

Reactor_id - Go to Referral and search for the Reactor_id. If present, use Morb_id to search Survclos. Example:

Errors	**Dbf: Test**	**Date: XX/XX/XXXX**	**Completed by: H. D. Clown**			Solutions
Reactor_id xxxxxxxxxx not in React	morb_id	>Survclos.Reacid				Place Reacid xxxxxxxxxx in Test
	xxxxxxxxxx	xxxxxxxxxx				

> - to file.field name xxxxxxxxxx - variable value

VISIT

This database contains information pertaining to public health clinic visits made by patients.

Problem: Patient_id not in Patient.dbf

The Visit record has 1 other variable that can possibly be used to find the solution to this problem.

Event_id - This method may be time consuming but is the best way to insure that linked files do not store a different, and possibly correct, Patient_id. The user must choose a linked database found using either the **Fields List** document or the **Key Linkages** document to search for the Event_id of the specific record. Possible databases with links: Fr, Morbrept, Reactor, and Test. Check only those databases that also have Patient_id stored as a field. If this error is found in multiple databases, the best solution may be to recreate the patient record in patient.

Problem: Address_id not in Address.dbf

The Visit record has 2 other variables that, if present, can possibly be used to find the solution to this problem. Only one solution will be presented for address fixes and it can be applied to any database that presents this problem. Depending on the number of these types of problems, the easiest solution to this problem may be to recreate the Address record in Address.dbf.

Patient_id - This problem may be resolved by using Epi-info Analysis to Read Address.dbf and select the Patient_id of the record with the errant Address_id. Then a listing should be made of all of that patient=s address_ids with As_of dates and/or date writs (add_dw). The date closest to the particular file should be used to replace the bad Address_id. The search can also be done in FDBU, setting the filter as described on page 6.

RECREATING SELECTED RECORDS

This section will cover the recreation of a few selected record types that the majority of referential integrity problems are associated with. Recreating a record should be done only after evaluating an error and deciding that the best course of action is to create a valid record for the missing criteria. Situations that may warrant record recreation include:

- the errant record or its linked records are valid and the recreation of another record is the

proper avenue to resolve the problem;

- the user is unsure of the validity of the record, but is hesitant to mark a record for deletion without further evidence that the record is invalid. (Examples: morbidity with a bad Patient_id or a Test record with a bad Address_id.)

Prior to recreating records in STD*MIS, the user should acquire as much information possible pertaining to the record that is being recreated. The recreated record may be as complete as a record done through routine data entry, or fairly incomplete housing only the minimum required data. Whenever possible, use complete, accurate data to recreate records. In some instances the user will have to use their best judgement. The Key ID is usually the variable that is Amissing@ in the seekkeys report and is the reason why the record is being recreated. Substitute the missing variable information where appropriate in the recreated record.

In FDBU open (<F2>) the database that the record is going to be recreated in, press <F5> to browse the record, and then press <CTRL> and the <Page Down> key to go to the bottom of the database. Next press the down arrow once. This will place the cursor in a blank record after the last complete record. In the upper right hand corner of the open file, the word A<new>@ will appear. Below are the individual database breakdowns for recreating records:

NOTE: Where noted that a particular value can be used do not put the quote marks into the data field.

MM - MONTHS HH - HOURS
DD - DAYS MM - MINUTES
YYYY - YEARS SS - SECONDS
Dates are divided by A/@s and time is divided by A:@s.

If the AUNKNOWN" values are not a part of the reference databases they must be added to the appropriate reference files.

ADDRESS Records

Variable Name	Mandatory Yes/No	Data/Definition/Format/Values
Address_id	Y	Key Id, retrieve from errant file. Use the Amissing@ value.
Patient_id	Y	Retrieve from errant file.
As_of	N	Can be blank. Date patient last known to live at this address.
Street_1	N	Can be blank. Actual street address of patient.
City	Y	Can input AUNKNOWN@ if actual city is unknown.
County	Y	Can input AUNKNOWN@ if actual county is unknown.

District	Y	Can input A98” or “99@ if actual District is unknown.
State	Y	Can input AUN@ if actual State is unknown.
Zip	Y	Can input A99999@ if actual Zip Code is unknown.
Census_tr	N	Can be blank.
Home_phone	N	Can be blank. If entered, place area code first with dashes separating.
Prov_fl	Y	Defaults to AF@ (false). If this was the provider=s address instead of the patient=s, AT@ (true) would be placed here.
Add_dw	Y	Date the Address record was created. MM/DD/YYYY
Add_tw	Y	Time the Address record was created. HH:MM:SS

FRB Records

Variable Name	Mandatory Yes/No	Data/Definition/Format/Values
Frb_id	Y	Key Id, retrieve from errant file. Use the Amissing@ value.
Dispo	N	Can be blank.*
Dispo_dt	N	Can be blank.*
Worker	Y	Can input A999@ if actual Worker is unknown.
Source_spr	N	Can be blank.
Post_cnsl	N	Can be blank.
Frb_dw	Y	Date the Frb record was created. MM/DD/YYYY
Frb_tw	Y	Time the Frb record was created. HH:MM:SS

*If the Field Record is to be closed, then a disposition and date are needed, else the record is open and can be closed as any other Fr would be closed within the system.

MORBREPT Records

Variable Name	Mandatory Yes/No	Data/Definition/Format/Values
Morb_id	Y	Key Id, retrieve from errant file. Use the Amissing@ value.
Patient_id	Y	Retrieve from linked file(s).

Address_id	Y	Retrieve from linked file(s).
Event_id	Y	Retrieve from linked file(s).
Origin	Y	Where did Morbidity originate. Codes C - Congenital Record F - Field Record I - Interview Record L - Lab (Test) Record M - Morbidity Report Record V - Clinic Visit Record
MMWR_week	Y	See Attached MMWR document.
MMWR_year	Y	See Attached MMWR document.
Rept_dt	Y	Date morbidity reported. MM/DD/YY
Pregnant	Y	AY@ for yes, AN@ for no, AU@ for unknown
Morb_age	Y	Age at time of morbidity report. A999" if unknown.
Named	Y	Named report. AT@ if named, AF@ if non-named.
Morb_entry	Y	Person entering Morbidity record. Three - Four characters.
Morb_dw	Y	Date the Morbidity record was created. MM/DD/YYYY
Morb_tw	Y	Time the Morbidity record was created. HH:MM:SS

PATIENT Records

Variable Name	Mandatory Yes/No	Data/definition/Format/Values
Patient_id	Y	Key Id, retrieve from errant file. Use the Amissing@ value.
Last_name	N	Can be blank.*
First_name	N	Can be blank.*
M_initial	N	Can be blank.
DOB	N	Can be blank.
SSN	N	Can be blank.
Sex	Y	A1" - Male, A2" - Female, A9" - Unknown.

Race	Y	A1" - American Indian A2" - Asian/Pacific Islander A3" - Black/African American A4" - White A8" - Other A9" - Unknown
Race_type	N	Can be blank.
Ethnicity	Y	A1" - Hispanic, A2" - Non-Hispanic, A9" - Unknown.
Med_rec_no	N	Can be blank.
Hars_id	N	Can be blank.
Archive	Y	Defaults to AF@.
Temp_dob	N	Can be blank.
Named	Y	Named report. AT@ if named, AF@ if non-named.
Pat_entry	Y	Person entering Morbidity record. Three - Four characters.
Pat_dw	Y	Date the Pateint record was created. MM/DD/YYYY
Pat_tw	Y	Time the Patient record was created. HH:MM:SS

* If Named='T', then one of these variables must contain data.

ATTACHMENTS

SAMPLE DOCUMENTATION SHEET
MMWR DOCUMENT

Errors	Dbf: Date: Completed by:					Solutions