**Background:**

When a user starts a transfusion in Bridge, results are sent across the Bridge Cloverleaf interface to Millennium and populate into a variety of event sets configured to store the results. Each of these results is associated to a ‘transaction id’ (parent\_entity\_id on the CLINICAL\_EVENT table) that signifies each of the results were sent within the same transaction. The results are also associated to a contributor system (Bridge), which identifies the system that sent the results.

Using the parent\_entity\_id and the contributor system, it is possible to associate all of the results sent across the interface to the unit number (W035812345678) in which they describe.

When a user ends the transfusion, the results are again sent across the interface to Millennium and populate into the appropriate event sets. Since this event is an entirely different transaction, the ‘transaction id’ (parent\_entity\_id) is *different* for the end transfusion results then it is for the start transfusion results. The **only** similar item contained in both transactions is the unit number.

**Issues:**

1. It is possible to join two CLINICAL\_EVENT tables together by the unit number, but the unit number is not unique.
   1. If any combination of the below scenarios happen (and potentially others not mentioned) the information pulled on the report by this join will be incorrect:
      1. Two split units with the same unit number on one patient
      2. Two split units with the same unit number on different patients within the time range defined by the report
      3. Two units with the same product number, but different ecodes on one patient
      4. Two units with the same product number, but different ecodes on two different patients within the range defined by the report
      5. Any unit information pulled from Blood Bank tables joined via the unit number

**Possible Resolutions:**

1. **Handle the scenario with error handling**

Make the join to the second CLINICAL\_EVENT table and determine the number of rows returned. If there is a single row returned, then the report should function fine – the unit number is unique for this particular scenario. Print the end transfusion dt/tm.

If more than one row is returned, then the report encountered one of the scenarios (i – v) above. The transfusion end dt/tm **cannot** be accurately defined on the report for either of the units. Several things could be done, but essentially they would involve populating all of the patient/unit information and noting on the patient that the transfusion end dt/tm must be looked up in the system (and manually written in?...whatever workflow you want).

**Pros:** Not too much additional coding

**Cons:** Users have to look up data in system

1. **Handle the scenario by modifying the Bridge configuration**

**In summary:**

Modify the Bridge interface code to concatenate the unit number and the division code (W031512345678 + A0) and use this as the unit number sent across the interface and recorded on the event set, create a new event set for the product ecode (E1234) and send the unit ecode across the interface as well (this second piece is optional based on your blood supplier).

**How it all works:**

The unit number is not unique itself; it needs the product ecode *and* the division code in order to make it unique. Bridge stores the division code and product barcode elements with the unit number in their database which creates a unique unit number in the patient’s record. That way, the unit number can be used multiple times and on different patients. If the product code is different but the unit number and division codes are the same, Bridge treats that as a unique unit as well.

When a unit is scanned, Bridge automatically parses out the division code from the product barcode and associates it to the unit number to ensure there is a unique number/division code combination in their database. This happens after the user scans the unit number from the donor bag and then scans the product ecode barcode. The product barcode contains the product code (ecode – E1234), the donation type (V) and the division code (A0). Since the database stores the product number associated with the division code, it is possible to concatenate the two together and send them as the unit number to Millennium.

As stated above, the unique unit number is not enough to identify the unit for those products who have the same unit number, division code but different ecodes *(this may not apply to your facility. If it does not, then we can ignore this last piece)*. This can be resolved by creating a new event set and passing the product ecode (E1234) as a result.

With the unique unit number and the product ecode, the join to the second CLINICAL\_EVENT table can be done with confidence that only a single row will return. **No modifications to any end-user workflow will result from these changes.**

Example:

Product code = “Aph RBC ACDA 500”

Product barcode = E1234VA0

Ecode = E1234

Product number = W038212345678

Division code = A0

Event set 1: Concatenate the product number + division code so “W038212345678A0” is sent

Event set 2: Donation type (“Volunteer” or whatever it is)

Event set 3: Either the product barcode or the ecode across the interface (“E1234VA0” or “E1234”)

Event set 4: Product code (“Aph RBC ACDA 500”)

**Pros:** The end transfusion dt/tm can accurately display on the report, the *actual* patient blood type and the blood type at the time of transfusion can display on the report **1**, increased report functionality (dispense dt/tm, dispense reason, difference in transfuse and dispense dt/tm, average transfusion dt/tm (from when unit is picked up to when unit is finished transfusing), cooler, courier, XM dt/tm, additional units still XM’ed to the patient…the list is infinite), no extra coding (if no additional BB fields are necessary)

**Cons:** Event set build, interface modification (Bridge should do this)

1. By having the unique unit number and product type, the *actual* patient blood type can be displayed on the report. Additional functionality can be added to compare the *actual* blood type with the *assumed* blood type (on this report or a separate report).

The blood type of the patient in Bridge, and thus the blood type written to the CLINICAL\_EVENT table during a transfusion, does not always reflect the *true* blood type of the patient. Unless otherwise changed, the blood type in Bridge is derived from the type of the first unit transfused to that patient. There are numerous examples of scenarios as to how a unit given to a patient doesn’t match their blood type (run out of proper blood type, emergency transfusion etc). Thus, the blood type that is documented on the CLINICAL\_EVENT table is the *assumed* blood type because it is purely the blood type at the time of transfusion.

Scenarios can arise when the patient is typed and their true blood type is determined. If a patient didn’t have a blood type in Millennium and were given two units of O Neg blood, their blood type would be O Neg in Bridge. After they are typed in Millennium, they are assigned a blood type (let’s say A Pos). If the patient is in need of additional units, they are most likely going to receive A Pos units. The problem is that A Pos is not a compatible blood type with O Neg, so when the nurse tries to transfuse an A Pos unit to an O Neg patient they will receive an error. The error message will differ based on your compatibility table and security settings, but hopefully you are not allowing nurses to override a patient’s blood type in Bridge…this should be a function reserved for the Blood Bank Supervisors.

In the scenario above, the patient’s *assumed* blood type in Bridge must be changed after the initial transfusion to the *actual* blood type in Millennium.

The report can pull both the ‘*assumed’* and ‘*actual’* blood types and a comparison can be made that will allow the Blood Bank Supervisor the ability to know whether they need to update the blood type in Bridge or not.