



[About Us](#) | 
 [Foundation](#) | 
 [Operations](#) | 
 [Regulatory](#) | 
 [Decision Support](#) | 
 [Directory](#) | 
 [Publications](#)

November 2005



[Click to return to Database Documentation](#)

## Quad-Based Data Validation Procedures - Program Flow

This document gives a brief description of each trigger that fires as records from agencies that place traps on a quad-based system are loaded into the database. The triggers are listed in the order that they fire.

### Quad-Based Trap Placement Data

Trigger Name: **TPLACE\_UPD\_BEF\_ROW**

Table: TEST\_PLACEMENT

Procedures Called: TdataPkg.Get\_GM\_Sequence\_ID

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tplace\_upd\_bef\_row.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tplace\_upd\_bef\_row.trg

**ACTION:** This trigger fires BEFORE INSERT, UPDATE, OR DELETE on the TEST\_PLACEMENT table for each row that is affected. Variables in the TdataPkg package are assigned values from the current row in the TEST\_PLACEMENT table. There are no changes made to the table data in order to avoid the dreaded 'mutating table' error. The variables will be referenced by the next trigger, TPLACE\_INS\_UPD\_AFT\_ROW.

Inserting:

The trigger fires and proceeds through this section when a row is inserted into the TEST\_PLACEMENT table. The function TdataPkg.GET\_GM\_SEQUENCE\_ID assigns the next value from the GMSEQ sequence to the ID column. Values are assigned to variables from the row being inserted.

If the record being inserted is from a West Virginia trapper, the variable M\_WV\_QUAD\_CODE may have a value. A lookup is done on the QUAD table to find the associated abbreviation. If a match is found, the abbreviation is assigned to the variable V\_LDR\_ABBREV. (West Virginia trappers once entered data using a set of numeric quad codes that are different from the quad abbreviations used by other agencies. The codes are listed in the WV\_QUAD\_CODE column of the QUAD table. *This method is not currently being used.*)

The next two 'IF' statements are here because we needed some creativity in loading data from the GPS records. There is limited space in the data entry fields, so we have had to make several of the character positions serve multiple purposes. The first 'IF' statement was used with Trimble Scoutmaster GPS receivers and is no longer used with the Magellan units. I left it in here in case the manufacturers of the Magellan units decide to add a similar feature to their units. The second 'IF' statement is still used to indicate that a trap was placed beyond the edge of the target circle.

First IF statement – (IF v\_gpsdata = 'L' THEN...) The column GPS\_DATA was added to the TEST\_PLACEMENT table when STS project personnel began to replace opscan forms with GPS units. A 'Y' is inserted into this column to indicate that the record was received from a GPS unit. The Trimble Scoutmasters included a feature that indicated the accuracy of the UTM coordinate. The valid entry types were 'ACU-LOCK', 'UNDER 30', and 'MANUAL'. (The Magellan GPS receivers do not have this feature.)

With the Scoutmaster, the database assumed that 30 'points' were recorded with every set of UTM coordinates. If the trapper stopped the GPS unit before reaching 30 points, he/she could enter the letter 'L' in the 14th position of the GPS record to indicate fewer than 30 points were recorded. The letter 'L' was loaded into the GPS\_DATA column. This trigger converts the letter 'L' to the letter 'Y' in the GPS\_DATA column and inserts 'UNDER 30' into the ENTRY\_TYPE column.

Second IF statement – (IF m\_outside\_target IS NULL AND v\_extravar = '8' THEN...) The column OUTSIDE\_TARGET in the TEST\_PLACEMENT table is used to indicate that a trapper placed a trap further than the desired distance from the target location. The trapper enters the letter 'B' in the position after the trap\_type code. (This position is also used to report the reason for omitting a site.) The letter 'B' is converted to the number 8 and is inserted into the EXTRA\_VAR column of the TEST\_PLACEMENT table. This 'IF' statement sets the OUTSIDE\_TARGET column to 'Y' and sets the EXTRA\_VAR column to NULL.

Updating:

If a row is being updated, the trigger deletes the rows from the ERRORS table with a value in the ROW\_ID column that matches the internal ROWID value of the row being updated in the TEST\_PLACEMENT table. The trigger also updates rows in the ERROR\_SUMMARY table and sets the DATE\_OUT column to the SYSDATE where the ID value matches the ID of the row being updated in the TEST\_PLACEMENT table.

Deleting:

If a row is being deleted from the TEST\_PLACEMENT table, the trigger deletes the appropriate rows from the ERRORS table based on the ROW\_ID value. It also updates the ERROR\_SUMMARY table and sets the DATE\_OUT value to the SYSDATE where the ID matches the ID of the row being updated in the TEST\_PLACEMENT table.

---

Trigger Name: **TPLACE\_INS\_UPD\_AFT\_ROW**

Table: TEST\_PLACEMENT

Procedures Called: TDATAPKG (package variables only)

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tplace\_ins\_upd\_aft\_row.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tplace\_ins\_upd\_aft\_row.trg

ACTION:

This trigger fires AFTER INSERT OR UPDATE on the TEST\_PLACEMENT table for each row that is affected. Its only action is to assign the internal row id value of the current row in the TEST\_PLACEMENT table to the row\_id variable in the TdataPkg package.

Trigger Name: **TPLACE\_INS\_UPD\_AFT**

Table: TEST\_PLACEMENT

Procedures Called: TDATAPKG (package variables only)

QUAD\_CONVERT

SITE\_VALID

DAY\_CHECK

FORMDATE\_LATER\_THAN\_SCAN

TRAPPER\_CONVERT

TRAPTYPE\_VALID

QUADSITE\_UNIQUE

GRIDUTMS\_UNIQUE

UTMS\_UNIQUE

OMIT\_WHY\_VALID

UTM\_QUAD\_CHECK

TRAP\_DISTANCE

TRAPS\_TOO\_CLOSE

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tplace\_ins\_upd\_aft.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tplace\_ins\_upd\_aft.trg

Action:

This trigger fires AFTER INSERT OR UPDATE on the TEST\_PLACEMENT table. A boolean variable, NO\_ERRORS, is set to true at the beginning of the trigger. If an error is encountered during the processing of the trigger NO\_ERRORS will be set to false and control continues with the next command. If at the end of the trigger NO\_ERRORS is still true, a row will be inserted into either the PLACEMENT or OMITTED\_SITES table based on the value of V\_TRAPTYPE. Otherwise, the row will remain in TEST\_PLACEMENT with the appropriate error codes entered into the ERRORS and ERROR\_SUMMARY tables.

The procedural flow of the trigger is as follows:

First, all variables are declared and assigned a data type. Two cursors are defined. The first (SITES\_CUR) selects a row from the SITE\_LOCATIONS table based on the values of the variables v\_usgs\_code and v\_site. The second cursor (FIRST\_COORDS\_ID) searches for previous entries in the FIRST\_COORDINATES table.

Following the first BEGIN statement, data values are assigned to the variables from the currently selected record.

The procedure QUAD\_CONVERT searches the QUAD table for the appropriate USGS\_CODE based on either the quad abbreviation or the West Virginia quad code.

If the site number either equals zero or is missing, the procedure SITE\_VALID is called to manage the error.

If there was an error with either the quad code or site number, the trigger raises the exception QUAD\_SITE\_ERROR and exits without performing any additional validations.

If both the quad code and site number are valid, the trigger continues by opening and fetching from the cursor SITES\_CUR. A record will be returned from the SITE\_LOCATIONS table if a matching quad and site is found for the values in the variables v\_usgs\_code and v\_site. If no matching record is found, the exception NO\_SITE\_DATA is raised, records are inserted into the ERRORS and ERROR\_SUMMARY tables, and processing ends.

If a record is found in the SITE\_LOCATIONS table with a matching quad and site value, processing continues with the following validations.

The variable M\_SCAN is filled with the current SYSTEM date that is loaded with the GPS record. The date will be formatted to be compatible with Oracle's default date format (DD-MON-YYYY).

(Note: The term SCAN is used because data was submitted using optical scan forms when the database was created in 1994. GPS units were first introduced into the project in 1996.)

The procedure DAY\_CHECK is called to check for a valid trap placement date. If there are no errors in the placement date, the procedure FORMDATE\_LATER\_THAN\_SCAN is called. This procedure checks for trap placement dates that are later than the data entry dates (m\_scan).

The procedure TRAPPER\_CONVERT is called to search for the trapper's initials in the PEOPLE table. Errors are logged if the trapper's initials are missing. The procedure also compares the agency value in the record from the SITE\_LOCATIONS table to the agency value in the record that is found in the PEOPLE table. If these values do not match an error is logged in the ERRORS and ERROR\_SUMMARY tables with constraint -20002.

The procedure TRAPTYPE\_VALID checks for an entry in the M\_TRAPTYPE field. If it is missing, errors are logged in the ERRORS and ERROR\_SUMMARY tables through the NULL\_TRAPTYPE exception. The code from the GPS record (D, M, or O) is converted as the record is inserted into the TEST\_PLACEMENT table. The valid values are DELTA, MILK CARTON, and OMIT. If any other code is entered in the

GPS record, it is inserted without any conversion into the trap\_type field in the TEST\_PLACEMENT table. If invalid values are found, the TRAPTYPE\_VALID procedure will insert records into the ERRORS and ERROR\_SUMMARY tables with the constraint 'CHK\_TESTPLACE\_TRAPTYPE'.

The procedure QUADSITE\_UNIQUE looks for an existing quad code and site number. If a match is located in either the PLACEMENT or the OMITTED\_SITES table, a 'duplicate site' error is logged in the ERRORS and ERROR\_SUMMARY tables.

The procedure GRIDUTMS\_UNIQUE looks for an existing grid node utm coordinate pair within the zone where the current record is located. If a match is located in either the PLACEMENT or the OMITTED\_SITES table, a 'duplicate grid node utms' error is logged in the ERRORS and ERROR\_SUMMARY tables.

The procedure UTMS\_UNIQUE looks for an existing utm coordinate pair within the zone where the current row is located. If a match is located in either the PLACEMENT or the OMITTED\_SITES table, a 'duplicate utms' error is logged in the ERRORS and ERROR\_SUMMARY tables.

If the value in v\_traptype is 'OMIT', the procedure OMIT\_WHY\_VALID checks for a valid value from the OMIT\_REASONS table. If the variable v\_why is null then the exception NULL\_OMIT\_REASON is raised. If the value in the variable v\_why is not found in the OMIT\_REASONS table, errors are logged with constraint –20018 through the exception INVALID\_OMIT\_REASON.

The procedure UTM\_QUAD\_CHECK performs several validations related to the utm coordinates. The procedure checks for missing values in the grid, utm\_east, and utm\_north columns. If there are no missing items, the procedure verifies that the grid node utm coordinates in the SITE\_LOCATIONS table lie within the limits for the specified quad. If the grid type is reported as 'random' the procedure checks the utm coordinates of the trap location instead of the grid node utm coordinates. Errors are logged in the ERRORS and ERROR\_SUMMARY tables. NOTE: There is a call to two procedures, ROUND\_UTME and ROUND\_UTMN. These procedures are no longer used but remain here in case they are needed in the future. These procedures were used previously to calculate the grid node coordinates when a site was not found in the SITE\_LOCATIONS table.

The procedure TRAP\_DISTANCE reports the distance of the trap from the grid node location. If the trap is located outside of the target circle, the procedure raises the exception OUTSIDE\_TARGET. The distance between the target grid node location and the actual trap location is calculated and stored in the variable 'trap\_dist' (meters). The distance between the trap location and the edge of the target circle is calculated and stored in the variable 'v\_outside\_target' (meters). This value will be inserted into the DISTANCE\_OUTSIDE column in the PLACEMENT table.

The procedure TRAPS\_TOO\_CLOSE checks the PLACEMENT table for any traps that have been placed within 100 meters of the current trap. The OMITTED\_SITES table is also checked for any records within 100 meters of the current trap. If any traps or omitted sites are located, a 'traps too close' error is logged in the ERRORS and ERROR\_SUMMARY tables.

A row is inserted into the FIRST\_COORDINATES table in order to store the original utm coordinates that are reported for each site. Since each record is to be stored the first time it enters the database, the cursor FIRST\_COORDS\_ID is used to search for an existing record. A row is inserted into the FIRST\_COORDINATES table if the ID is not found; otherwise, control continues to the next step.

The error flag NO\_ERRORS is checked at this point. If one or more errors were encountered the NO\_ERRORS variable is now FALSE. No further action will be taken. However, if NO\_ERRORS has remained TRUE, the trigger will perform two additional actions. First, the SITE\_LOCATIONS table will be updated. A 'Y' will be entered into the PLACED column to indicate that the site has been addressed.

Second, a row is inserted into either the PLACEMENT or OMITTED\_SITES table. The trigger checks the variable V\_TRAPTYPE. If the value is 'OMIT', a row is inserted into the OMITTED\_SITES table. If the value is either 'DELTA' or 'MILK CARTON', a row is inserted into the PLACEMENT table.

---

Trigger Name: **OMIT\_INSERT\_AFT**

Table: OMITTED\_SITES

Procedures Called: No procedures

Script File:

Sceolaun -> C:\oracle\trap\_triggers\omit\_insert\_aft.trg

Mothsbane -> D:\trap\_files\trap\_triggers\omit\_insert\_aft.trg

Action:

This trigger fires when a row is INSERTED into the OMITTED\_SITES table. It deletes a row from the TEST\_PLACEMENT table if there is an ID value that matches the ID of the row being inserted into OMITTED\_SITES. By coding this action into a trigger, it ensures that the row is inserted into the OMITTED\_SITES table before the row is deleted from TEST\_PLACEMENT.

---

Trigger Name: **PLACE\_INSERT\_AFT**

Table: PLACEMENT

Procedures Called: No procedures

Script File:

Sceolaun -> C:\oracle\trap\_triggers\place\_insert\_aft.trg

Mothsbane -> D:\trap\_files\trap\_triggers\place\_insert\_aft.trg

Action:

This trigger fires when a row is INSERTED into the PLACEMENT table. It deletes a row from the TEST\_PLACEMENT table if there is an ID value that matches the ID of the row being inserted into PLACEMENT. By coding this action into a trigger, it ensures that the row is inserted into the PLACEMENT table before the row is deleted from TEST\_PLACEMENT.

---

### Quad-Based Trap Inspection Data

Trigger Name: **TINSPECT\_UPD\_BEF\_ROW**

Table: TEST\_INSPECTION

Procedures Called: TdataPkg.Get\_GM\_Sequence\_ID

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tinspect\_upd\_bef\_row.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tinspect\_upd\_bef\_row.trg

ACTION:

This trigger fires BEFORE INSERT, UPDATE, or DELETE on the TEST\_INSPECTION table. This trigger fires for each row that is affected. Variables in the TdataPkg package are assigned values from the current row in the TEST\_INSPECTION table. There are no changes made to the table data in order to avoid the dreaded 'mutating table' error. The variables will be referenced by the next trigger, TINSPECT\_INS\_UPD\_AFT\_ROW.

Inserting:

The trigger fires and proceeds through this section when a row is inserted into the TEST\_INSPECTION table. The function TdataPkg.GET\_GM\_SEQUENCE\_ID assigns the next value from the GMSEQ sequence to the ID column. Values are assigned to variables from the row being inserted.

If the record being inserted is from a West Virginia trapper, the variable M\_WV\_QUAD\_CODE will have a value. A lookup is done on the QUAD table to find the associated abbreviation. If a match is found, the abbreviation is assigned to the variable V\_LDR\_ABBREV. (West Virginia trappers enter data using a set of numeric quad codes that are different from the quad abbreviations used by other agencies. The codes are listed in the WV\_QUAD\_CODE column of the QUAD table.)

The next three 'IF' statements are here because we needed some creativity in loading data from the GPS records. There is limited space in the data entry fields, so we have had to make several of the character positions serve multiple purposes. The first 'IF' statement was used with Trimble Scoutmaster GPS receivers and is no longer used with the Magellan units. I left it in here in case the manufacturers of the Magellan units decide to add a similar feature to their units.

First IF statement – (IF v\_gpsdata = 'L' THEN...) The column GPS\_DATA was added to the TEST\_PLACEMENT table when STS project personnel began to replace opscan forms with GPS units. A 'Y' is inserted into this column to indicate that the record was received from a GPS unit. The Trimble Scoutmasters included a feature that indicated the accuracy of the UTM coordinate. The valid entry types were 'ACU-LOCK', 'UNDER 30', and 'MANUAL'. (The Magellan GPS receivers do not have this feature.) With the Scoutmaster, the database assumed that 30 'points' were recorded with every set of UTM coordinates. If the trapper stopped the GPS unit before reaching 30 points, he/she could enter the letter 'L' in the 14th position of the GPS record to indicate fewer than 30 points were recorded. The letter 'L' was loaded into the GPS\_DATA column. This trigger converts the letter 'L' to the letter 'Y' in the GPS\_DATA column and inserts 'UNDER 30' into the ENTRY\_TYPE column.

The second 'IF' statement (ELSIF v\_gpsdata IS NULL THEN...) simply ensures that the GPS\_DATA column contains a 'Y'.

The third 'IF' statement (ELSIF TO\_NUMBER(v\_gpsdata) >= 0 THEN...) allows us to use the last two characters in the GPS record (positions 19 and 20) for two purposes. A trapper can record a catch value during a routine trap inspection, or a supervisor can enter a QC failure code during a 'less-than-perfect' quality control inspection. SQL\*Loader will not allow one position in the input file to be designated as character data in one case and numeric data in another. So, we assign that position (20) to a character data type and convert to a numeric data type when necessary. There are three possible scenarios:

1. A trapper enters a routine trap catch. The trapper is required to enter three digits for the catch value. The first two digits (positions 18 and 19) are inserted into the CATCH column, and the third digit (position 20) is inserted into the GPS\_DATA column. This trigger multiplies the CATCH value by 10, then converts the value in GPS\_DATA to a numeric value and adds it to the CATCH value. This final value will be inserted into the CATCH column in the INSPECTION table.
2. A supervisor enters a passing QC inspection. He/she enters one digit into the CATCH column (position 18) and the letter 'P' into the FIELD\_CHECK column (position 19). Position 20 is left blank, and a null value is assigned to the GPS\_DATA column.
3. A supervisor enters a failing QC inspection. He/she enters one digit into the CATCH column (position 18), the letter 'F' into the FIELD\_CHECK column (position 19), and a QC failure code into the QC\_FAIL column (position 20).

Updating:

If a row is being updated, the trigger deletes the rows from the ERRORS table with a value in the ROW\_ID column that matches the internal ROWID value of the row being updated in the TEST\_INSPECTION table. The trigger also updates rows in the ERROR\_SUMMARY table and sets the DATE\_OUT column to the SYSDATE where the ID value matches the ID of the row being updated in the TEST\_INSPECTION table.

Deleting:

If a row is being deleted from the TEST\_INSPECTION table, the trigger deletes the appropriate rows from the ERRORS table based on the ROW\_ID value. It also updates the ERROR\_SUMMARY table and sets the DATE\_OUT value to the SYSDATE where the ID matches the ID of the row being updated in the TEST\_INSPECTION table.

---

Trigger Name: **TINSPECT\_INS\_UPD\_AFT\_ROW**

Table: TEST\_INSPECTION

Procedures Called: TDATAPKG (variables only)

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tinspect\_ins\_upd\_aft\_row.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tinspect\_ins\_upd\_aft\_row.trg

ACTION:

This trigger fires AFTER INSERT OR UPDATE on the TEST\_INSPECTION table for each row that is affected. Its only action is to assign the internal row id value of the current row in the TEST\_INSPECTION table to the row\_id variable in the TdataPkg package.

---

Trigger Name: **TINSPECT\_INS\_UPD\_AFT**

Table: TEST\_INSPECTION

Procedures Called: TDATAPKG  
 DAY\_CHECK  
 FORMDATE\_LATER\_THAN\_SCAN  
 QUAD\_CONVERT  
 SITE\_VALID  
 OMITTED\_SITE  
 PLACED\_SITE  
 TRAPPER\_VALID  
 INSPECT\_UNIQUE  
 UTMS\_UNIQUE  
 UTM\_QUAD\_CHECK  
 TRAP\_DISTANCE  
 TRAPS\_TOO\_CLOSE  
 QC\_VALID  
 VISIT\_FINAL  
 VISIT\_VALID  
 TRAP\_CONDITION  
 CATCH\_VALID  
 PLACEMENT\_UPD

Script File:

Sceolaun -> C:\oracle\trap\_triggers\tinspect\_ins\_upd\_aft.trg

Mothsbane -> D:\trap\_files\trap\_triggers\tinspect\_ins\_upd\_aft.trg

ACTION:

This trigger fires AFTER INSERT OR UPDATE on the TEST\_INSPECTION table. A boolean variable, NO\_ERRORS, is set to true at the beginning of the trigger. If an error is encountered during the processing of the trigger NO\_ERRORS will be set to false and control continues with the next command. If at the end of the trigger NO\_ERRORS is still true, a row will be inserted into the INSPECTION table. Otherwise, the row will remain in the TEST\_INSPECTION table with the appropriate error codes inserted into the ERRORS and ERROR\_SUMMARY tables.

The procedural flow of the trigger is as follows:

First, all variables are declared and assigned a data type. Two cursors are defined. The first (SITES\_CUR) selects the ID and AGENCY from the SITE\_LOCATIONS table and the UTM\_ZONE from the QUAD table for records matching the values of the variables v\_usgs\_code and v\_site. The second cursor (TOT\_CATCH) selects the total\_catch value from the PLACEMENT table where the quad and site match the values of the v\_usgs\_code and m\_site variables.

Following the first BEGIN statement, data values are assigned to the variables from the currently selected record. If the table is being updated, the ID value is assigned from the o\_id variable in the TdataPkg package.

The variable M\_SCAN is filled with the current SYSTEM date that is loaded with the GPS record. The date will be formatted to be compatible with Oracle's default date format (DD-MON-YYYY).

(Note: The term SCAN is used because data was submitted using optical scan forms when the database was created in 1994. GPS units were first introduced into the project in 1996.)

The procedure DAY\_CHECK is called to check for a valid trap inspection date. If there are no errors in the inspection date, the procedure FORMDATE\_LATER\_THAN\_SCAN is called. This procedure checks for trap inspection dates that are later than the data entry dates (m\_scan).

The procedure QUAD\_CONVERT searches the QUAD table for the appropriate USGS\_CODE based on either the quad abbreviation in the variable v\_ldr\_abbrev or the West Virginia quad code in v\_wv\_quad\_code.

If the site number either equals zero or is missing, the procedure SITE\_VALID is called to manage the error.

Next open the SITES\_CUR cursor and select the following values into variables: SITE\_LOCATIONS.ID, SITE\_LOCATIONS.AGENCY, QUAD.ZONE. The cursor is closed if no record is returned; otherwise, the PREDETERMINED\_SITE variable is set to true.

If there were no errors with either the quad code or site number, then the OMITTED\_SITE procedure is called. This procedure searches the OMITTED\_SITES table for a record matching the quad and site values in the variables v\_usgs\_code and v\_site. There should be no inspections since there are no traps placed at omitted sites. An error will be generated if a row is found in the OMITTED\_SITES table. If there are no matches in the OMITTED\_SITE table, the PLACED\_SITE procedure is called. This procedure searches for a row in the PLACEMENT table based on the values in the v\_usgs\_code and v\_site variables. A row with a matching quad and site value must exist in the PLACEMENT table before a row can be inserted into the INSPECTION table. An error will be generated if no match is found in the PLACEMENT table.

The procedure TRAPPER\_VALID is called to search for the trapper's initials in the PEOPLE table. Errors are logged if the trapper's initials are missing. The procedure also compares the agency value in the record from the SITE\_LOCATIONS table to the agency value in the record that is found in the PEOPLE table. If these values do not match an error is logged in the ERRORS and ERROR\_SUMMARY tables with constraint -20002.

If there are no quad or site errors, the procedure INSPECT\_UNIQUE checks for an existing record in the INSPECTION table. Only one record with the same quad, site, inspection date, and field check value may be entered. Errors are logged in the ERRORS and ERROR\_SUMMARY tables if a record exists with matching values for all four columns.

The next section was developed for use with the Trimble Scoutmaster GPS receivers. The Scoutmaster units enabled the trapper to record UTM coordinates with various levels of accuracy. The most accurate data entry type is ACU-LOCK, followed by UNDER 30, MANUAL, and OPSCAN. The variable M\_REPLACE is originally set to FALSE. The variable M\_ENTRYTYPE is assigned the value of the current ENTRY\_TYPE column from the TEST\_INSPECTION table. The variable N\_ENTRYTYPE is assigned the value of the ENTRY\_TYPE column from the row with a matching quad and site value in the PLACEMENT table. The variable M\_REPLACE will be set to TRUE if the value of M\_ENTRYTYPE is considered to be more accurate than the value of N\_ENTRYTYPE. If M\_REPLACE is TRUE, the UTM coordinates in the current TEST\_INSPECTION record are checked more closely. The following procedures are called: utms\_unique, utm\_quad\_check, trap\_distance, and traps\_too\_close. If there are no errors, the UTM coordinates in the PLACEMENT table are replaced with the UTM

coordinates from the current TEST\_INSPECTION record. The ENTRY\_TYPE column in the PLACEMENT record will be updated with the value from the TEST\_INSPECTION record (M\_ENTRYTYPE). The valid value for ENTRY\_TYPE when entering data from the Magellan units is MAGELLAN. The variable M\_REPLACE will remain FALSE, so processing will continue with the call to the QC\_VALID procedure.

The QC\_VALID procedure checks for valid values in the FIELD\_CHECK and QC\_FAIL columns. The valid values for the FIELD\_CHECK column are F, N, P, or NULL. If the value of the FIELD\_CHECK column is F, then the QC\_FAIL column must contain a valid value from the QC\_FAIL\_REASONS table. Errors are logged in the ERRORS and ERROR\_SUMMARY tables if these conditions are not met.

If the value of the M\_VISIT variable is 'FINAL', the VISIT\_FINAL procedure is called to check for two conditions. The VISIT column must not be NULL, and there must exist only one record for each quad and site with a value of 'FINAL' in the VISIT column and a value of 'N' in the FIELD\_CHECK column. A trapper may remove a trap only one time; however, a supervisor may check the site multiple times to be sure the trapper removed the trap. There may exist multiple records with the same quad and site with a value of 'FINAL' in the VISIT column and a value of 'P' or 'F' in the FIELD\_CHECK column to indicate a QC inspection.

The procedure VISIT\_VALID checks for a valid value in the VISIT column (V\_VISIT). Valid entries are 'MIDSEASON' or 'FINAL', and the column may not be null. The procedure also checks the value in the DAY column (V\_DAY) to ensure that all midseason inspections occur prior to a final inspection. It is assumed that the trap is removed from the field at the earliest final visit. Errors are logged in the ERRORS and ERROR\_SUMMARY tables.

The next section regarding DEPTH and EXTRA VARIABLES has been commented out. This section was used for a short time to help trappers record moth catches in areas where there were very high gypsy moth populations. The trapper would install milk carton traps and measure the depth of the moths in the traps instead of counting individual moths. The formula could then be applied to convert the depth to a catch value. This method of recording trap catch values is no longer used. I kept it in the trigger in case it needs to be revived at some point.

The procedure TRAP\_CONDITION checks for a valid value in the CONDITION column (M\_TRAP\_COND). The column may not be null, and the valid values are: GOOD, DAMAGED, INACCESSIBLE, MISSING. The database assumes that a trapper can physically touch a trap and look inside the trap for moths when the condition is reported as 'GOOD' or 'DAMAGED'. If a trapper returns to a trap site for an inspection and is unable to either get access to the exact location or is unable to find the trap, the condition should be entered as 'MISSING' or 'INACCESSIBLE'. The condition should be entered as 'MISSING' if the trap is so severely damaged that moths cannot be counted. Errors are logged in the ERRORS and ERROR\_SUMMARY tables if the CONDITION column is NULL or if any other values are entered.

If there are no errors in the CONDITION column, then the procedure CATCH\_VALID is called. This procedure checks for the following situations:

1. If the value of CONDITION (V\_TRAP\_COND) is 'MISSING' or 'INACCESSIBLE', then CATCH (M\_CATCH) must be NULL. (The trapper cannot count moths if he/she cannot look inside of the trap.)
2. If the value of CONDITION (V\_TRAP\_COND) is 'GOOD' or 'DAMAGED', then CATCH (M\_CATCH) must be greater than or equal to zero. (The trapper must report a moth catch of zero or greater.)

Errors are logged in the ERRORS and ERROR\_SUMMARY tables.

If at this point the error flag (NO\_ERRORS) remains true, then each of the column values are assigned to a table record. This record is inserted into the INSPECTION table.

The procedure PLACEMENT\_UPD is called to update the parent row in the PLACEMENT table. The first IF statement will be ignored since M\_REPLACE will be FALSE while the Magellan GPS units are in use. If M\_REPLACE is TRUE, the UPDATE statement will set the UTM coordinates to the values from the inspection record (M\_UTME and M\_UTMN). It will also set the ENTRY\_TYPE column to the value in M\_ENTRYTYPE from the inspection record, and it will update the DISTANCE\_OUTSIDE column with the figure that was calculated using the UTM coordinates in the inspection record (V\_OUTSIDE\_TARGET).

The TOTAL\_CATCH value is updated in the PLACEMENT table based on the previous TOTAL\_CATCH value (S\_CATCH) in the PLACEMENT record and the CATCH value in the new inspection record (V\_CATCH). The TOTAL\_CATCH value in the PLACEMENT table is indicated as follows:

-2 = no inspection (default value when row inserted into PLACEMENT table)

-1 = site inspected by trap was missing/inaccessible at all inspections

0+ = total moth catch for the site

The TOTAL\_CATCH column value will be updated as follows:

1. Change TOTAL\_CATCH from -2 to -1 only if this is the first inspection and the trap is missing/inaccessible.
2. Recalculate TOTAL\_CATCH only if the catch value is > 0. If TOTAL\_CATCH = -1 or -2, set TOTAL\_CATCH to 0 before adding new catch value. (A null catch value indicates a missing/inaccessible trap.)

---

Trigger Name: **INSPECT\_INSERT\_AFT**

Table: INSPECTION

Procedures Called: None

Script File:

Sceolaun -> C:\oracle\trap\_triggers\inspect\_insert\_aft\_trg.sql

Mothsbane -> D:\trap\_files\trap\_triggers\inspect\_insert\_aft\_trg.sql

ACTION:

This trigger fires AFTER a row is INSERTED into the INSPECTION table. It deletes a row from the TEST\_INSPECTION table if there is an ID value that matches the ID of the row being inserted into the INSPECTION table.

---

Procedure Name: DAY\_CHECK

Called By: PLACEMENT\_AFT trigger

INSPECT\_AFT\_UPD trigger

Procedures Called: NONE

## Action:

This procedure checks for a valid trap placement/inspection date and formats the date for entry into the appropriate table. If the date is null, the DAY\_MISSING exception is raised. If the date exists but is not a proper date, the INVALID\_DAY exception is raised. If the date is valid, it is formatted for entry into the appropriate table and stored in the variable V\_DAY.

Procedure Name: FORMDATE\_LATER\_THAN\_SCAN

Called By: PLACEMENT\_AFT

INSPECT\_AFT\_UPD

Procedures Called: NONE

## Actions:

The date entered on the form or into the GPS receiver is stored in the variable V\_DAY. If V\_DAY occurs later than the date the forms are scanned (C\_SCAN), then the exception DATE\_LATER\_THAN\_SCAN is raised.

Procedure Name: QUAD\_CONVERT

Called By: PLACEMENT\_AFT

INSPECT\_AFT\_UPD

Procedures Called: NONE

## Action:

This procedure checks for a valid quad abbreviation or valid WV quad code. The NULL\_QUAD exception is raised if both values are null. If either the quad abbreviation or the WV quad code is present, the cursor QUAD\_PROC searches for the corresponding USGS code in the QUAD table. If this is a valid code, the USGS code will be stored in the variable V\_USGS\_CODE. If no match is found, the exception INVALID\_ABBREVIATION is raised. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables.

Procedure Name: SITE\_VALID

Called By: PLACEMENT\_AFT

INSPECT\_AFT\_UPD

Procedures Called: NONE

## Action:

The SITE\_VALID procedure looks for a site number greater than zero. If there are no errors, the site number is stored in the variable V\_SITE. The exception NULL\_SITE is raised if the site number is missing, and the exception INVALID\_SITE is raised if the site number equals zero. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables.

Procedure Name: TRAPPER\_CONVERT

Called By: PLACEMENT\_AFT

Procedures Called: NONE

## Action:

The cursor TRAPPER\_CUR selects the ID from the PEOPLE column based on the trapper's initials in the variable V\_TRAPPER\_INIT. If the initials are located in the PEOPLE table, the trapper's ID is stored in the variable M\_TRAPPER\_ID. If the trapper's initials are missing, the exception NULL\_TRAPPER is raised. The exception INVALID\_TRAPPER is raised if the initials are not found in the PEOPLE table. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables.

Procedure Name: TRAPTYPE\_VALID

Called By: PLACEMENT\_AFT

Procedures Called: NONE

## Action:

If the trap type is missing, the exception NULL\_TRAPTYPE is raised and errors are written to the ERRORS and ERROR\_SUMMARY tables. Otherwise, the trap type is stored in the variable V\_TRAPTYPE.

Procedure Name: QUADSITE\_UNIQUE

Called By: PLACEMENT\_AFT

Procedures Called: NONE

## Action:

The cursor OMIT\_CUR searches for an existing site in the OMITTED\_SITES table based on the values stored in the variables V\_USGS\_CODE and V\_SITE. If an existing record is located, the exception DUP\_OMITTED\_QUADSITE is raised. The cursor PLACE\_CUR performs the same search on the PLACEMENT table. The exception DUP\_PLACEMENT\_QUADSITE is raised if an existing record is located. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If no records are located control returns to the calling procedure, PLACEMENT\_AFT.

Procedure Name: UTMS\_UNIQUE

Called By: PLACEMENT\_AFT

Procedures Called: NONE

Action:

The cursor OMIT\_CUR searches for an existing site in the OMITTED\_SITES table based on the utm coordinates stored in the variables M\_UTME and M\_UTMN. If an existing record is located, the exception DUP\_OMITTED\_UTMS is raised. The cursor PLACE\_CUR performs the same search on the PLACEMENT table. The exception DUP\_PLACEMENT\_UTMS is raised if an existing record is located. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If no records are located, control returns to the calling procedure, PLACEMENT\_AFT.

---

Procedure Name: GRIDUTMS\_UNIQUE

Called By: PLACEMENT\_AFT

Procedures Called: NONE

Action:

The cursor OMIT\_CUR searches for an existing site in the OMITTED\_SITES table based on the grid node utm coordinates stored in the variables M\_UTME\_GRID and M\_UTMN\_GRID. If an existing record is located, the exception DUP\_OMITTED\_GRIDNODE is raised. The cursor PLACE\_CUR performs the same search on the PLACEMENT table. The exception DUP\_PLACEMENT\_GRIDNODE is raised if an existing record is located. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If no records are located, control returns to the calling procedure, PLACEMENT\_AFT.

---

Procedure Name: UTM\_QUAD\_CHECK

Called By: PLACEMENT\_AFT

Procedures Called: ROUND\_UTME

ROUND\_UTMN

Action:

This procedure checks the utm coordinates to ensure that the site is located within the reported quad. If the grid type is reported as either 'Random' or 'Other', the utm coordinates are checked. Otherwise, the grid node utms are checked. If the grid type is missing, the exception NULL\_GRID is raised. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If either one of the utm coordinates is missing, one of the following exceptions is raised - NULL\_UTME or NULL\_UTMN. If both of the utm coordinates are missing the exception NULL\_UTMS is raised. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If necessary, the procedures ROUND\_UTME and ROUND\_UTMN are called in order to calculate the grid node coordinates. If the BOOLEAN variables Y\_UTME and Y\_UTMN are both true, the coordinates are within the quad boundaries. However, if either Y\_UTME or Y\_UTMN is false, the exception OUTSIDE\_QUAD is raised. Errors are inserted into the ERRORS and ERROR\_SUMMARY tables.

---

Procedure Name: ROUND\_UTME

Called By: UTM\_QUAD\_CHECK

Procedures Called: NONE

Action:

This procedure checks the grid type stored in the variable V\_GRID. If the grid type equals 'RANDOM' or 'OTHER', the grid node utm easting will be equal to the utm easting stored in the variable M\_UTME. Otherwise, the following formula is used to calculate the grid node easting value:

$$m\_utm\_grid := TRUNC((m\_utme/v\_roundist)+0.5)*v\_roundist$$

The rounding distance (V\_ROUNDIST) is stored in the ROUNDING\_DISTANCE column of the GRID table and is selected based on the grid type (V\_GRID). The grid node utm easting coordinate M\_UTME\_GRID is returned to the UTM\_QUAD\_CHECK procedure.

---

Procedure Name: ROUND\_UTMN

Called By: UTM\_QUAD\_CHECK

Procedures Called: NONE

Action:

This procedure checks the grid type stored in the variable V\_GRID. If the grid type equals 'RANDOM' or 'OTHER', the grid node utm northing will be equal to the utm northing stored in the variable M\_UTMN. Otherwise, the following formula is used to calculate the grid node northing value:

$$m\_utmn\_grid := TRUNC((m\_utmn/v\_roundist)+0.5)*v\_roundist$$

The rounding distance (V\_ROUNDIST) is stored in the ROUNDING\_DISTANCE column of the GRID table and is selected based on the grid type (V\_GRID). The grid node utm northing coordinate M\_UTMN\_GRID is returned to the UTM\_QUAD\_CHECK procedure.

---

Procedure Name: TRAP\_DISTANCE

Called By: PLACEMENT\_AFT

Procedures Called: NONE

Action:

The cursor TARGET\_CUR selects from the GRID table the distance to be used as the radius of the target circle. If the requested grid type is not found in the GRID table, the exception INVALID\_GRID is raised. If the grid is okay, the absolute value of the distance between the grid node and location coordinates is stored in the variables B\_UTME and B\_UTMN. The BOOLEAN variables X\_UTME and X\_UTMN will flag coordinates that are outside the target circle. If either X\_UTME or X\_UTMN are false, the exception OUTSIDE\_TARGET is raised. The



distance between the trap location and the grid node is stored in the variable TRAP\_DIST. The variable V\_OUTSIDE\_TARGET contains the distance that the trap is located outside the target circle.

Procedure Name: TRAPS\_TOO\_CLOSE

Called By: PLACEMENT\_AFT

Procedures Called: NONE

Action:

This procedure checks for traps that are placed too close to the current location. The cursor DISTANCE\_CUR selects rows from the PLACEMENT table where the utm coordinates are fewer than 72 meters from the current record. Using the distance formula,  $d = \sqrt{\text{POWER}((x2-x1),2) + \text{POWER}((y2-y1),2)}$ , the easting AND northing coordinates of another trap must be at least 72 meters from another trap. The exception TRAPS\_TOO\_CLOSE is raised if a row is found, and errors are inserted into the ERRORS and ERROR\_SUMMARY tables. If no rows are selected, control returns to the calling procedure.