

**Exor to AgileAssets Maintenance and PMS Interface**

**Interface Design**

**Version 1.1**

**September 2013**



1.0 Introduction 3

2.0 High Level Requirements 3

3.0 Road Network Information 4

3.1 Full List of Routes and Addition and Removal of Routes 4

3.2 Route Spatial Representation 5

4.0 Event Information 6

4.1 Updating the Location of ‘Event’ Linear References 6

5.0 Asset Information 9

5.1 Updating Asset Information 9

6.0 Interface Execution 13

7.0 Reports 13

7.1 Report Processing 16

8.0 Documentation Requirements 28

9.0 Assumptions 28

10.0 Conclusion 28

11.0 Appendix I 29

Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changed by | Notes |
| February 2012 | 0.1 | RE | Initial Revision |
| May 13, 2013 | 0.2 | JM | Addressed KYTC concerns |
| September 10 2013 | 1.1 | RE | Changes to address KY comments |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Reference Documents

Site Visit to KYTC v0.3.docx – Site Visit Report, Richard Ellis, June 2012

Exor to AgileAssets PMS interface.docx - Design of an interface between the Exor product and the AgileAssets PMS, 2007.

Exor to AgileAssets interface Scope and Requirements.docx – Requirements document for the interface between the KYTC Exor system and the AgileAssets System.

# Introduction

In May of 2012, Richard Ellis of Bentley Systems visited KYTC to review the current implementation of the Exor product, and to plan future improvements to it. During this visit it was identified that the creation of an Exor to AgileAssets data exchange interface offered a significant opportunity to improve the quality of information held in the AgileAssets system and reduce the effort currently expended keeping some aspects of the two systems coordinated. This will improve the quality of road information held by the state and reduce the cost of duplicated data entry.

KYTC and Bentley Systems have established a project to undertake the scope and requirements analysis, and to create a design for this interface. That analysis resulted in a Scope and Requirements document that has been agreed between Exor and KYTC.

This report is a software design document that defines software that meets the requirements identified. It will be used to ensure everyone has a common understanding of the software this project will create. This report will then be used to define a further project to create and implement the software.

# High Level Requirements

The Scope and Requirements document established that the main objective of this project was to create software that could replicate what KYTC currently does manually to update the AgileAssets System with road network information. This includes:

* Provision of a full list of routes
* Addition and removal of routes
* Updating route spatial representation
* Updating the location of ‘Event’ linear references and spatial locations
* Updating asset information

The interface will be flexible enough that if the asset information that will be shared between Exor and the AgileAssets system should change, these changes can be implemented by changes in the configuration without needing changes in the interface software.

This interface should allow both the Maintenance and the PMS systems (currently both implemented using software from AgileAssets) to have location of their business objects on the road network locations maintained without the need to make manual changes to the road network information held within the AgileAssets system.

# Road Network Information

## Full List of Routes and Addition and Removal of Routes

The Exor system must provide a current version of the road network information so that the AgileAssets system can take a current set of network data to establish the two systems with an initial, common version of the road network, and to re-synchronize the network at any time in the future.

To make this data available, tables will be created in the Exor database and access provided to the AgileAssets system. These views allow the identification of new routes and new length on existing routes. Gaps in the linear referencing system are identified. Alternative linear referencing methods and concurrent routes are not used by KYTC.

Route Sections Table defines the extent of each route:

View XAA\_ROUTE

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Null? | Data Type | Comment |
| ROUTE\_ID | N | NUMBER | Route ID |
| ROUTE\_NAME | N | VARCHAR2 (30 Byte) | Route Name |
| ROUTE\_TYPE | N | VARCHAR2 (4 Byte) | Network type |
| OFFSET\_FROM | N | NUMBER (22,4) | From Milepoint |
| OFFSET\_TO | N | NUMBER (22,4) | To Milepoint |
| EFFECTIVE\_DATE | N | DATE | Effective date |

Bentley will construct this view using the following logic:

|  |  |
| --- | --- |
| Column | Derivation |
| ROUTE\_ID | The NE\_ID of the Route |
| ROUTE\_NAME | The Unique of the Route |
| ROUTE\_TYPE | The Network type of the route. This will always be ‘RT’ (County Route). |
| OFFSET\_FROM | Minimum measure on the route section |
| OFFSET\_TO | Maximum Measure on the route section |
| EFFECTIVE DATE | Maximum Date on the members of the route |

Notes:

Route Section is a section of route where the roadway is continuous and not broken by a distance break. This table will hold multiple sections of the same route if the route has gaps in the linear referencing. Distance breaks of length zero still mark the beginning and end of a section. Bentley will then construct a process that recreates the XAA\_ROUTE table.

KYTC does not have concurrent routes so no tables are planned to support concurrent routes.

The AgileAssets system also requires a table of spatial data. This table will hold one geometry per route and have the following format:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Null?** | **Data Type** | **Comment** |
| ROUTE\_ID | N | NUMBER(38) | Route ID |
| GEOLOC | N | MDSYS.SDO\_GEOMETRY, GTYPE 3306 | Shape as an Oracle spatial geometry with measures on the vertices |

Bentley will construct this view using the following logic:

|  |  |
| --- | --- |
| **Column** | **Derivation** |
| ROUTE\_ID | The NE\_ID of the Route |
| GEOLOC | The geometry of the route |

## Route Spatial Representation

Every change in spatial representation of the route must be tracked so that changes in shape not normally treated as an edit are still captured and reported. For example, if a datum is ‘reshaped’ this is not tracked as a network change because it is considered data improvement or correction, not a physical change in location of the assets. The AgileAssets System still needs to be made aware of these changes.

A trigger will be added to the route spatial table, so that on insert or delete, a row will be added to the XAA\_SPATIAL\_AUDIT table. On update, two rows will be added to the XAA\_SPATIAL\_AUDIT table indicating a delete and an add. The AgileAssets system can use this information to retrieve the appropriate shapes from the route spatial table discussed above.

The XAA\_SPATIAL\_AUDIT table will have the following structure:

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Null? | Data Type | Comment |
| ROUTE\_ID | N | Number(38) | Route ID |
| GEOLOC | N | MDSYS.SDO\_GEOMETRY | Shape as an Oracle spatial geometry |
| OPERATION | N | VARCHAR2(6) | Either ADD or DELETE |
| OP\_DATE | N | Date | Date the operation occurred |
| EFF\_DATE | N | Date | Start date of the Geometry |
| END\_DATE | Y | Date | End date of the Geometry |
|  |  |  |  |

Bentley will construct this table via a trigger using the following logic:

|  |  |
| --- | --- |
| Column | Derivation |
| ROUTE\_ID | The NE\_ID of the Route |
| GEOLOC | The geometry of the route that is being changed. For a delete operation this will be the old geometry and for a add operation this will be the new geometry |
| OPERATION | If the trigger is fired by insert this will be ‘ADD’ or if the trigger is fired by delete this will be ‘DELETE’ |
| OP\_DATE | SYSDATE of the trigger firing |
| EFF\_DATE | Start Date of the Geometry being added or removed in the route spatial table |
| END\_DATE | End date of the Geometry being added or removed in the route spatial table. This will usually, but not exclusively, be null for geometry being added. |

# Event Information

## Updating the Location of ‘Event’ Linear References

The AgileAssets System needs to know if any of the location of objects stored in the AgileAssets System has changed due to a network operation on the road network. To communicate this, the AgileAssets System will publish a set of historic network locations and a date of those locations. EXOR will respond with the current route locations for those objects.

The format of both the data received from the AgileAssets System and the data returned will be:

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Null? | Data Type | Comment |
| HISTORIC\_DATE | N | Date | Date of the locations coming from the AgileAssets System |
| LOC\_IDENT | N | INTEGER | AgileAssets Location ID # |
| ROUTE\_NAME | N | VARCHAR2 (30 Byte) | Route Name **in the form of an Exor System UNIQUE** |
| OFFSET\_FROM | N | NUMBER (22,4) | From Milepoint |
| OFFSET\_TO | N | NUMBER (22,4) | To Milepoint |
| SOURCE\_TABLE | N | VARCHAR2 (32 Byte) | Raw Data Table Name -  Informational Item that lists our source data table name |
| NEW\_DATE | N | Date | Date of the locations coming from the Exor System  (filled by Exor) |
| NEW\_ROUTE\_NAME | Y | VARCHAR2 (30 Byte) | New Route Name (filled by EXOR) |
| NEW\_OFFSET\_FROM | Y | NUMBER (22,4) | New From Milepoint (filled by EXOR) |
| NEW\_OFFSET\_TO | Y | NUMBER (22,4) | New To Milepoint (filled by EXOR) |
| PROCESS\_MSG | Y | VARCHAR(100) | Messages and Errors produced during the process. Identified possible errors are:  “invalid old location”  “Route location closed”. |

The data will be read from one table and written to another. The AgileAssets System will write to a table named **XAA\_LOC\_IDENT**. Exor will read from this table and write to a table named **XAA\_NET\_REF**.

If the old location has become two new locations separated by a section of road (that may be zero length such as a zero length distance break) then two, or more, rows will be returned to list all new locations. For example, the middle mile of a 3 mile route is re-aligned to be 0.2 mile shorter, so that milepoints from 1 and 2 become a new datum from 1 to 1.8 and milepoints 2 through 3 become 1.8 to 2.8. Locations that span the realignment would return two (or more) records: one for the section before the realignment, giving the new network location at that point and one for the section now realigned, with null values for the route, start and end measure, and a possibly a third, for the section after the realignment (if it exists) with the new location of this section.

When the historic location no longer exists (because the road has been closed) , null will be returned for the NEW\_ROUTE\_NAME, NEW\_OFFSET\_FROM and NEW\_OFFSET\_TO and an error inserted into the PROCESS\_MSG column with a message similar to “Route location closed”.

If the Historic location didn’t exist at the date specified then NEW\_ROUTE\_NAME, NEW\_OFFSET\_FROM and NEW\_OFFSET\_TO will be null and a process message similar to “invalid old location” will be placed in the PROCESS\_MSG column.

In some rare cases it may not be possible to establish a new location. In these cases a process message similar to “New location unavailable” will be placed in the PROCESS\_MSG column. These cases are created when the Exor software is able to track locations across a road network editing operation. There are no known instances where this will occur and with a weekly refresh of the network information this is highly unlikely to affect the operation of the interface. Testing and preventing these rare instances will cost a great deal more than is sensible.

To generate the new information, Bentley will construct a process to generate the data required. The interface will perform the following processing:

* The process will read each line of the XAA\_LOC\_IDENT table and identify if the route has been altered since the date specified. This can be done quickly by checking the XAA\_ROUTE table constructed above. If the date of the most recent change in the route is prior to the date specified in the XAA\_LOC\_IDENT then the route and location have not changed and the information can be written to the XAA\_NET\_REF table.
* If the route does not exist in the XAA\_ROUTE table then the only information written to the XAA\_NET\_REF table is the information provided and the PROCESS\_MSG “Route location does not exist”
* If the date of the most recent change in the route is after the date specified in the XAA\_LOC\_IDENT table, then the datum that existed at that measure along the route specified is determined. If that datum has not been end dated since the date specified then the location along the datum is determined, and the location of that datum in the route system is determined and that location information can be written to the XAA\_NET\_REF table.
* If the route does not include the measure specified at the date specified, then the information written to the XAA\_NET\_REF table is the information provided and the PROCESS\_MSG “Invalid old location”.
* If the route does not include the measure specified at the current date, then the information written to the XAA\_NET\_REF table is the information provided and the PROCESS\_MSG “Route Location Closed”.
* If location is unchanged then the information written to the XAA\_NET\_REF table is the information provided and the PROCESS\_MSG “Route Location Unchanged”.
* If the datum at that location has been end dated, then the history tables of Exor are used to determine what happened to that location. Several operations may have occurred, meaning that several datum may have been involved. The NM\_ELEMENTS\_HISTORY table can be used to track these changes.

NM\_ELEMENT\_HISTORY holds a record of all network operations:

|  |  |
| --- | --- |
| Operation | NM\_ELEMENT\_HISTORY.  NEH\_OPERATION |
| split | S |
| merge | M |
| replace | R |
| close | C |
| reclassify | N |
| recalibrate | B |
| shift | H |
| edit | E |
| reverse | V |

After each operation, a new NE\_ID is created. If multiple operations have occurred then each needs to be processed in order until the final datum location is determined and it can be processed into a route location and written to the XAA\_NET\_REF table.

A procedure will be created for each operation. The input parameters for each operation will be the row from NM\_ELEMENT\_HISTORY and each will return a replacement NE\_ID and a measure along that datum. An exception will be the ‘close’ operation. If a datum is closed then the network location no longer exists and the only information written to the XAA\_NET\_REF table is the information provided and the PROCESS\_MSG “Route location closed”.

The processes used will be:

Split - the datum has been split into two parts. The location will exist on one or other of them at a new measure

Merge - the datum has been merged with another datum and a new datum created. The measure may have been changed if the datum was the second datum and a new measure will need to be calculated

Replace - the datum has been replaced with another datum and a new datum created, no change in measure has occurred.

Close - discussed above, no processing required.

Reclassify - the datum has been replaced with another datum and a new datum created, no change in measure has occurred.

Recalibrate - the length has changed and a new datum created. The measure may have changed depending on the range of the reclassify measure.

Shift - Similar to reclassify, shift will change the location of some items, and therefore may change the measure, and create a new datum.

Edit - no change in measure, but a new datum has been created.

Reverse - A new datum has been created and the measure is now the distance from the end of the datum.

A program will loop through each operation performed on the network to determine what datum currently holds the location specified and what measure on the datum the location is.

A final procedure will convert the datum location to a current route location and the location will be written to the XAA\_NET\_REF table.

# Asset Information

## Updating Asset Information

Initially there are nine sets of asset information that must be communicated from Exor to the AgileAssets system. Details of the specific types of asset information to be communicated are defined in Appendix 1, but the nine types are:

AL - Auxiliary Lane

FS – Functional Class

LN – Lanes

RA – Adequacy Rating

RW – Right of way Width

SH – Shoulders

SL – Speed Limit

SS – State System

TF – Traffic Count Information

The interface will initially be configured to communicate this set of asset information to the AgileAssets system, but other asset information must be able to be added to the list by changing the configuration of the software and without the need for changes to the application software.

To achieve this, Bentley will create a series of tables. The name of the table will be defined by the interface metadata.

All asset tables will have the same five columns, and then the specific asset attribute columns required. The fixed columns will be:

Name Null? Type

-------------------- ------ -------------------------

ROUTE No VARCHAR2(50)

ROUTE\_TYPE No VARCHAR2(4)

FROM\_POINT No NUMBER

TO\_POINT No NUMBER

ASSET\_ID No NUMBER

The flexible attribute columns will then be whatever is specified for the interface configuration.

For example, The Auxiliary Lane table would have the following format if it was configured as per Appendix I

Name Null? Type

------------------ ------ -------------------------

ROUTE No VARCHAR2(50)

ROUTE\_TYPE No VARCHAR2(50)

FROM\_POINT No NUMBER

TO\_POINT No NUMBER

ASSET\_ID No NUMBER

AUXLANE NUMBER

AUXLNWID VARCHAR2(50)

AUXSURF VARCHAR2(50)

As many tables as required will be created by a process that reads the metadata and generates the tables.

To support the generation of the asset data, two metadata tables will be required. The first table will specify what asset types are exported and what tables the data will reside in. The second will specify what attributes are exported.

Asset type metadata will be held in a table XAA\_ASSET\_TYPE with the following columns:

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Null? | Data Type | Comment |
| ASSET\_TYPE | N | VARCHAR2(4) | The four letter code for the asset type. |
| TABLE\_NAME | N | VARCHAR2(30) | The name of the table that will hold the resulting data |
| ROUTE\_TYPE | N | VARCHAR2(4) | The group type that the asset should be reported on. For KYTC this will always be ‘RT’ |

Asset attribute type metadata will be held in a table XAA\_ASSET\_ATTRIB with the following columns:

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Null? | Data Type | Comment |
| ASSET\_TYPE | N | VARCHAR2(4) | The four letter code for the asset type. |
| COLUMN\_SEQ | N | NUMBER | Order of the columns in the table |
| COLUMN\_NAME | N | VARCHAR2(30) | Name of the column name in the table defined by XAA\_ASSET\_TYPE.TABLE\_NAME |
| COLUMN\_DATATYPE | N | VARCHAR2(60) | Defines the data type of the column. i.e. NUMBER, VARCHAR2(30), etc |
| COLUMN\_  DERIVATION | N | VARCHAR2(80) | An asset ‘VIEW\_COLUMN’ attribute value or a function using a ‘VIEW\_COLUMN’ attribute or a simple function or a literal value. This information is used in a select statement from the view V\_NM\_XXXX\_NW where XXXX is the four letter asset code. |

The table name can be used by more than one asset type provided the COLUMN\_SEQ, COLUMN\_NAME and COLUMN\_DATATYPE columns are kept identical for both assets. The COLUMN\_DERIVATION may be different.

Each time the data is generated for the interface the following processing will take place:

* All tables listed in the XAA\_ASSET\_TYPE.TABLE\_NAME will be dropped and recreated. If the table does not exist this will not cause an error.

Tables listed in the XAA\_ASSET\_TYPE.TABLE\_NAME will be created using the information in the COLUMN\_SEQ, COLUMN\_NAME and COLUMN\_DATATYPE columns of the XAA\_ASSET\_ATTRIB table and the standard columns. If the table is used by more than one asset type then it will be dropped and recreated more than once.

* The tables will be populated using the current information in the Exor database as follows:

|  |  |
| --- | --- |
| **Column** | **Derivation** |
| ROUTE | NE\_UNIQUE of the route the asset is located on |
| ROUTE\_TYPE | Route type of the route the asset is located on. For KYTC this would be ‘RT’ |
| FROM\_POINT | The minimum measure along the route for the asset |
| TO\_POINT | The maximum measure along the route for the asset |
| ASSET\_ID | The primary key (IIT\_PRIMARY\_KEY) of the asset |
| <flexible attributes> | The remainder of the columns will be filled by the attribute information defined by the XAA\_ASSET\_ATTRIB table. |

The flexible attributes will be populated using a dynamic SQL string constructed from the definition in the tables XAA\_ASSET\_TYPE and XAA\_ASSET\_ATTRIB.

# Interface Execution

In the sections above, various parts of the interface have been defined. All of these components must be implemented to complete the interface. There is a small amount of interdependency, so the order is important.

The interface will be implemented as a series of procedures and functions, wrapped in a database package and the tables and views will be created to support those. The interface will be executed either by a job that runs it at preset times, or manually, or both. The system administrator at KYTC will be responsible for creating the job based on administration documentation provided by Bentley. Bentley will provide a user interface for the process to be run from inside the Exor application, which will be particularly useful for testing.

All of the interface components will reside in the Exor system owner schema, but all output and input tables will reside in a separate schema. This will enable the data read from and sent to the AgileAssets system to be separated from core Exor data.

A data link will need to be forged between the AgileAssets system and the Exor system. The direction and mode of this link have yet to be confirmed but by using a separate schema and views reading data across database link(s), this can be defined as required and offsite development and testing will be more practical.

The interface components will be run in the following order:

* Full List of Routes and Addition and Removal of Routes
* Route Spatial Representation
* Updating the Location of ‘Event’ Linear References
* Updating Asset Information

Reporting can be run as a separate operation but should be run at the same time as the interface.

# Reports

A report is required so that staff can review what road network changes have occurred and check that the appropriate changes have been applied. KYTC also needs to be able to issue length change notices to various parts of the organization so that other systems can be manually updated. This report needs to list all changes to the length of Routes caused by network edits that add or remove datum from the network.

A single report to address both purposes will be derived from detecting changes to the members of the linear group routes. Any insert or update will be checked to see if it affected the linear referencing of the Route (many types of changes will not affect its referencing). If it does, then a record needs to be written to a table along with the ‘start date’ and ‘modified date’. This table can then be used to produce this report by specifying a date range.

Road datum are automatically added to the route groups whenever they are created (and are removed whenever the datum is closed) because auto-inclusion has been configured for the datum type. (Auto-inclusion is a standard, but optional, function of the Exor software). This means that the Exor software will ensure that:

* All datum are members of a linear group
* A datum cannot be moved from one group to another without a ‘Reclassify’ operation on the datum

A length change will occur whenever:

**A new section of road is added to a Route** (i.e. a datum is added to a linear group) because:

A new Route is created

The Route is extended (at either the beginning or end)

Route realignment

**A section of road is removed from a Route** (i.e. a datum is end dated in a linear group, and not replaced) because:

The Route is closed in its entirety

The Route is shortened at either end

Realignment of the Route

**A section of road is removed from one Route and placed in another Route**. This will create the same events as a combination of both the above cases.

**A gap is created in a Route**. This would result in a non zero length distance break being added.

**A gap is removed from a Route.** This would result in a non zero length distance break being removed.

**A section of road is recalibrated** (i.e. a datum is recalibrated) because it was found to be of the wrong length and an error correction was required. This would mean the length of datum would change. No change would occur in the route until the route was rescaled (in a separate operation) where the record would be end dated and a new record created with the same ID. Routes should always be rescaled after a network operation, but because recalibrate does not make a change in the route directly, care needs to be taken to ensure these changes are not missed. KYTC currently check this business rule with a weekly report.

Operations that might occur that would not cause a length change to be issued are:

* the merging of two road datum (Merge)
* the splitting of two road datum (Split)
* changes to any groups that are not of the type being tracked including non linear groups
* the bulk update of asset locations (such as ‘Shift’)
* a reshaping of the network geometry
* the replacement of a road datum for an identical one (a ‘replace’ operation). This may be done by a user to make a snapshot in history of a road datum before another operation (such as reshape).

In all cases, the length change notice must report the impact of the change on the linear referencing of the route, not just the operation itself. The impact of the change on the Route may not be apparent until the entire network operation has been completed, and this may entail several commit points.

For example, if a datum of road is recalibrated to be shorter, two commits are required for the operation to be completed and the change finalized on the route. The first is after the recalibration action is complete and a commit is executed to confirm the change to the length of the element. Then the user must rescale the route, or add a distance break to the route, so that the linear referencing along the Route is valid. For this reason, the tracking of length changes cannot be generated by simply tracking each individual change as they are committed, but must be collected at the end of each editing period when it is assumed the network editing operation is complete.

The system is not able to detect that two operations are the result of one larger network change. For example, if a realignment operation results in several new datum of road being added, then the table of ‘length changes’ may contain several changes which all neighbor each other that could be combined before they are issued. This may mean that the Length Change Report should be reviewed and modified manually before it is issued.

History is kept in the NM\_MEMBERS\_ALL table with start and end dates. In order for the length change notices to be correct, all operations must be completed to ensure the correct information is reported. If they are not completed, the change will still be reported, but its impact on the route may be erroneous. For example, if a datum is added, but the rescale is not executed until the next reporting period, the additional length will be recorded at the measure held in the first reporting period, and not updated after the rescale.

In order for the report to not repeat the listing of changes that have occurred previously, the reporting process must collect the data required between reporting periods. To enable this, the reporting process will record a ‘reporting date’. When the reporting process is run, only changes since this reporting date will be considered so as to enable the reporting of changes since that date and to avoid repeating the reporting of previous changes.

Routes should always be rescaled after a series of network operations, even if the operation made no apparent changes to the Route. For example, a recalibrate of a datum at the end of a Route or on a Route of only one road datum may not appear to require a rescale, but without it, the linear referencing of the route will not change and the effect on the total route length may not be detected.

The processes used to create the Length Change reporting objects are illustrated in the diagram

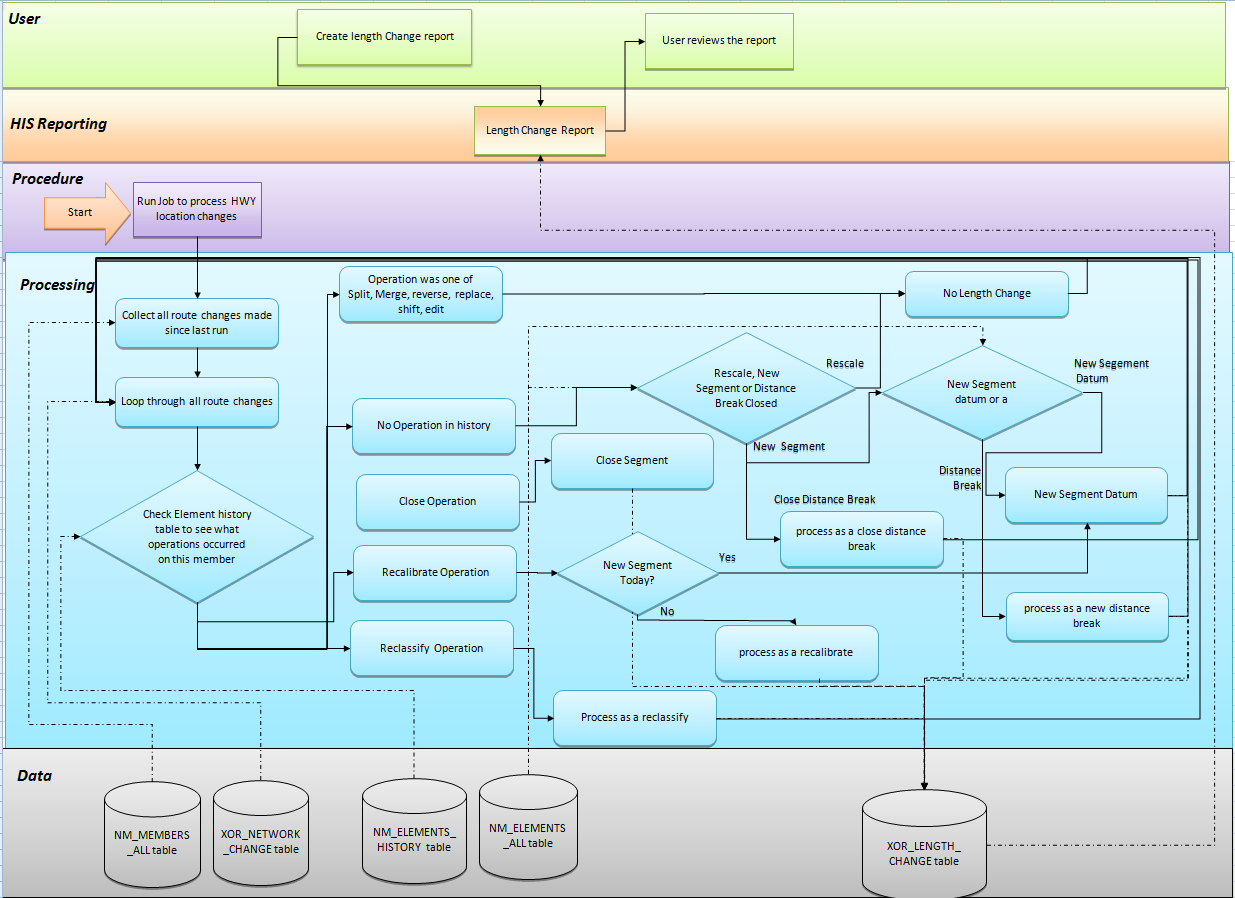


Figure 1 - Detailed Process Flow – Length Change Reporting Objects

## Report Processing

In the following sections, each component described in the detail process flow for the report is described in detail.

### NM\_MEMBERS\_ALL Table

The NM\_MEMBERS\_ALL table holds the relationship of road datum and distance breaks to network groups (NM\_TYPE = ‘G’) and assets (NM\_TYPE = ‘I’). It holds the ID of the road datum or distance break (NM\_NE\_ID\_OF), of the group (NM\_NE\_ID\_IN), where it is located in the Route (NM\_SLK and NM\_END\_SLK), when the record was modified (NM\_DATE\_MODIFIED), when the record became effective (NM\_START\_DATE) and when the record stopped being effective (NM\_END DATE).

### XOR\_NETWORK\_CHANGE Table

The XOR\_NETWORK\_CHANGE table holds the dates of when the data for the report was generated and is used to process all network changes between that date and the current date.

### NM\_ELEMENTS\_ALL Table

The NM\_ELEMENTS\_ALL table holds a record for every network element (datum, distance break, group or group of groups. This process only queries records that are datum which have an NE\_TYPE of ‘S’ and Distance breaks which have a NE\_TYPE of ‘D’. All records have a unique ID in NE\_UNIQUE and a unique primary key in NE\_ID.

### NM\_ELEMENTS\_HISTORY Table

A record of network operations is kept in the table NM\_ELEMENT\_HISTORY if any of the following network operations have occurred:

|  |  |
| --- | --- |
| Operation | NM\_ELEMENT\_HISTORY.  NEH\_OPERATION |
| split | S |
| merge | M |
| replace | R |
| close | C |
| reclassify | N |
| recalibrate | B |
| shift | H |
| edit | E |
| reverse | V |

### XOR\_LENGTH\_CHANGE Table

The changes in route length will be collected and stored in the table XOR\_LENGTH\_CHANGE and the length change report can then be generated from this table. The table will have the following definition:

|  |  |
| --- | --- |
| **XOR\_LENGTH\_CHANGE Column** | **Description** |
| CHANGE\_ID | A unique ID of the change. This number will be retrieved from a new sequence called XOR\_LENG\_CHANGE\_SEQ |
| CHANGE\_DATE | Date the change was made. |
| EFFECTIVE DATE | The start date of length added or the end date of length removed |
| DATUM\_ID | Primary key (NE\_ID) of the datum |
| DATUM\_ UNIQUE | Unique ID of the datum being removed |
| DATUM\_LENGTH | Length of the datum affected. For RECALIBRATE operations this is the length **after** the operation is complete. |
| DATUM\_TYPE | Type of length added or removed. It will be ‘DATUM’ for all datum and ‘DISTANCE BREAK’ for all distance breaks. |
| OPERATION | One of ADDED, CLOSED, RECALIBRATED LONGER or RECALIBRATED SHORTER, ADDED RECLASSIFY or CLOSED RECLASSIFIY, GEOMETRY |
| OLD\_BEGIN\_MEASURE | Begin MP of the datum before the operation, null if the section is being added |
| OLD\_END\_MEASURE | End MP of the datum before the operation, null if the section is being added |
| NEW\_BEGIN\_MEASURE | Begin MP of the datum after the operation, null if the section is being removed. |
| NEW\_END\_MEASURE | End MP of the datum after the operation, null if the section is being removed. |
| CHANGE\_START\_MEASURE | The measure of the change along the Route, the source of this value will change depending on what operation has taken place but is generally  (Old\_End\_Measure – Old\_Begin\_Measure) |
| CHANGE\_END\_MEASURE | The measure of the change along the Route, the source of this value will change depending on what operation has taken place but is generally  (New\_End\_Measure – New\_Begin\_Measure) |
| MILEAGE\_CHANGE | Change in mileage (positive or negative)  (Change\_End\_Measure – Change\_New\_Measure) |
| ROUTE\_ID | Primary Key (NE\_ID) of the route affected |
| ROUTE\_UNIQUE | Unique ID of the route affected |
| ROUTE\_NAME | Name of the route from the group description field |

### Job to Process Route Location Changes

After each network editing period the process will be run, either triggered by a DBMS job or started manually. This procedure will then trigger the processes to create the list of length changes and store them in the XOR\_LENGTH\_CHANGE table. This procedure starts with the ‘Collect all Route Network Changes‘ process.

### Collect all Network Changes

The process will scan the NM\_MEMBERS\_ALL table for records added or updated (end dated) since the last run that relate to RT groups. This can be detected from the NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE column using the date stored in the XOR\_LENGTH\_CHANGE table. These records are then used in other processes starting with ‘Loop through all Network Changes’

### Loop through All Network Changes

Every NM\_MEMBERS\_ALL record that was modified needs to be analyzed to see if it was the result of an operation that affected the linear referencing of the route. This is tested in other processes starting with ‘Check Element History Table’

### Check Element History Table

For each NM\_MEMBERS\_ALL record, this process looks for a record in the NM\_ELEMENT\_HISTORY table (where NM\_NE\_ID\_OF = NEH\_NE\_ID\_NEW) and based on what is found, another process is run determined by the logic below.

If the operation was one of Split, Merge, Replace, Shift, Edit or Reverse (where NM\_ELEMENT\_HISTORY.NEH\_OPERATION is in ‘S’, ‘M’, ‘R’, ‘H’, ‘E’ or ‘V’ ) then no length change is required and the process ‘**No Length Change’** is processed next.

If there is no record in the NM\_ELEMENT\_HISTORY table then the process **‘No Operation in History’** is processed next.

If there is a record in the NM\_ELEMENT\_HISTORY where NM\_ELEMENT\_HISTORY.NEH\_OPERATION of ‘B’ (recalibrate) and the date of the NM\_ELEMENT\_HISTORY.NEH\_ACTIONED\_DATE equals the NM\_MODIFED\_DATE of the NM\_MEMBERS\_ALL record then the process **‘Recalibrate Operation’** is processed next.

If there is a record in the NM\_ELEMENT\_HISTORY table where NM\_ELEMENT\_HISTORY.NEH\_OPERATION of ‘R’ (Reclassified) and the date of the NM\_ELEMENT\_HISTORY.NEH\_ACTIONED\_DATE equals the NM\_MODIFED\_DATE of the NM\_MEMBERS\_ALL record then the process **‘Reclassify Operation’** is processed next.

If there is a record in the NM\_ELEMENT\_HISTORY where NM\_ELEMENT\_HISTORY.NEH\_OPERATION of ‘C’ (Close) and the date of the NM\_ELEMENT\_HISTORY equals the NM\_MODIFED\_DATE of the NM\_MEMBERS\_ALL record then the process **‘Close Operation’** is processed next.

It is possible that the datum may occur in the NM\_ELEMENT\_HISTORY more than once because the recalibrate operation will not change the NE\_ID of the datum. This is because:

* + The recalibrate of a single datum on the same day will result in only one member record so this can be ignored and treated as a single recalibrate.
  + A single datum may be recalibrated and split or merged, or shifted or edited or reversed. All of these should be processed as a **‘recalibrate’**
  + A single datum may be recalibrated and closed but will result in only one member record. This record needs to be processed as both a recalibrate and as a close.
  + If a datum is created and recalibrated on the same day, it will be detected as a recalibrate. The recalibrate process must therefore check if it was created on the same day and process it accordingly.

### Operation of Split, Merge, Reverse, Replace, Shift, Edit

When the operation is one of Split, Merge, Replace, Shift, Edit or Reverse (where NM\_ELEMENT\_HISTORY.NEH\_OPERATIONin ( ‘S’, ‘M’, ‘R’, ‘H’, ‘E’, ‘V’) and the operation type is not ‘B’ (recalibrate) or ‘C’ (close) or ‘R’ (reclassify) then no record is required and the process **‘No Length Change’** is processed next.

### No Length Change

If it is determined that this change in the NM\_MEMBERS\_ALL table resulted in no length change then processing of the current NM\_MEMBERS\_ALL record can halt and the next in the queue processed. The next process should be ‘Loop through All HWY Network Changes’

### No Operation in History

If it’s not in the NM\_ELEMENT\_HISTORY table, then it was either added today or rescaled today or a distance break was removed today. This is tested by the process ‘Rescale or New Datum’

### Rescale or New Datum or Distance Break Closed.

If there is a record in NM\_ELEMENTS\_ALL (where NM\_NE\_ID\_OF = NE\_ID) and NM\_ELEMENTS\_ALL.NE\_START\_DATE = NM\_MEMBERS\_ALL.NM\_START\_DATE, and the NM\_ELEMENTS\_ALL.NE\_TYPE = ‘S’ then the elements was added to the route today.

Different processing is required for new datum or distance breaks. If the NM\_ELEMENTS\_ALL.NE\_TYPE indicates the section was ‘S’ (datum), then the process ‘**New Datum’** is processed next. If the NM\_ELEMENTS\_ALL.NE\_TYPE indicates the section was ‘D’ (Distance Break) then the process ‘**New Distance Break’** is processed next.

If there is not a record in NM\_ELEMENTS\_ALL (where NM\_NE\_ID\_OF = NE\_ID) and NM\_ELEMENTS\_ALL.NE\_START\_DATE = NM\_MEMBERS\_ALL.NM\_START\_DATE then this change might be the result of the removal of a distance break. To confirm this, NM\_ELEMENTS\_ALL will be queried to confirm this record refers to a distance break where the NM\_MEMBERS\_ALL record with NM\_START\_DATE equal to the NM\_END\_DATE of the member record being processed, and the NM\_NE\_ID\_OF = NM\_NE\_ID\_OF of the member record being processed and NM\_NE\_ID\_IN = NM\_NE\_ID\_IN of the member record being processed. If one does exist then the distance break was not closed, only rescaled, or closed as the result of another operation that is being tracked If one does not exist then the distance break was closed and the process Close Distance Break needs to be processed next. If not then this member change is the result of a rescale or closed as the result of another operation that is being tracked and no length change occurred and the process ‘No Length Change’ is processed next.

### New Datum

If the section is a new datum then a record must be inserted into the XOR\_LENGTH\_CHANGE table as follows:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | Start date of the NM\_MEMBER record (NM\_MEMBER.NM\_start\_date |
| DATUM\_ID | NM\_MEMBER.Nm\_ne\_id\_of |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_ unique (NM\_MEMBERS\_ALL.Nm\_ne\_id\_of) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL.Nm\_ne\_id\_of) |
| DATUM\_TYPE | DATUM |
| OPERATION | ADDED |
| OLD\_BEGIN\_MEASURE | Null |
| OLD\_END\_MEASURE | Null |
| NEW\_BEGIN\_MEASURE | NM\_MEMBERS\_ALL.NM\_SLK |
| NEW\_END\_MEASURE | NM\_MEMBERS\_ALL.NM\_END\_SLK |
| CHANGE\_START\_MEASURE | NM\_MEMBERS\_ALL.NM\_SLK |
| CHANGE\_END\_MEASURE | NM\_MEMBERS\_ALL.NM\_END\_SLK |
| MILEAGE\_CHANGE | NM\_END\_SLK - NM\_SLK |
| ROUTE\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_IN |
| ROUTE\_UNIQUE | NM3NET.GET\_NE\_UNIQUE(NM\_NE\_ID\_IN) |
| ROUTE\_NAME | NM3NET.GET\_NE\_DESCR(ROUTE\_ID) |

A subsequent operation may or may not mean that this NM\_ELEMENTS\_ALL.NM\_END\_DATE is null.

The next change in the Route in the queue should now be processed. The next process should be ‘Loop through All Network Changes’.

### New Distance break

If the section is a new Distance Break then a record must be inserted into the XOR\_LENGTH\_CHANGE table as follows:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | Start date of the NM\_MEMBER record (NM\_MEMBER.NM\_start\_date |
| DATUM\_ID | NM\_MEMBER.Nm\_ne\_id\_of |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_ unique (NM\_MEMBERS\_ALL.Nm\_ne\_id\_of) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL.Nm\_ne\_id\_of) |
| DATUM\_TYPE | DISTANCE BREAK |
| OPERATION | ADDED |
| OLD\_BEGIN\_MEASURE | Null |
| OLD\_END\_MEASURE | Null |
| NEW\_BEGIN\_MEASURE | NM\_MEMBERS\_ALL.NM\_SLK |
| NEW\_END\_MEASURE | NM\_MEMBERS\_ALL.NM\_END\_SLK |
| CHANGE\_START\_MEASURE | NM\_MEMBERS\_ALL.NM\_SLK |
| CHANGE\_END\_MEASURE | NM\_MEMBERS\_ALL.NM\_END\_SLK |
| MILEAGE\_CHANGE | NM\_END\_SLK - NM\_SLK |
| ROUTE\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_IN |
| ROUTE\_UNIQUE | NM3NET.GET\_NE\_UNIQUE(NM\_NE\_ID\_IN) |
| ROUTE\_NAME | NM3NET.GET\_NE\_DESCR(ROUTE\_ID) |

A subsequent operation may or may not mean that this NM\_ELEMENTS\_ALL.NM\_END\_DATE is null.

The next change in the Route in the queue should now be processed. The next process should be ‘Loop through All Network Changes’.

### Close Operation

When the operation is one of ‘C’ (close) then a record is required and the process ‘Close Datum’ is processed next.

### Close Datum

When a road datum has been closed a record must be inserted into the XOR\_LENGTH\_CHANGE table as follows:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | End date of the NM\_MEMBERS\_ALL record (NM\_MEMBERS\_ALL.NM\_END\_DATE |
| DATUM\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_OF |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_TYPE | DATUM |
| OPERATION | CLOSED |
| OLD\_BEGIN\_MEASURE | NM\_SLK |
| OLD\_END\_MEASURE | NM\_END\_SLK |
| NEW\_BEGIN\_MEASURE | Null |
| NEW\_END\_MEASURE | Null |
| CHANGE\_START\_MEASURE | NM\_SLK |
| CHANGE\_END\_MEASURE | NM\_END\_SLK |
| MILEAGE\_CHANGE | NM\_SLK - NM\_END\_SLK |
| ROUTE\_ID | NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN |
| ROUTE\_UNIQUE | Nm3net.get\_ne\_unique(NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN) |
| ROUTE\_NAME | Nm3net.get\_ne\_descr(route\_ID) |

### Close Distance Break

When a Distance Break has been closed a record must be inserted into the XOR\_LENGTH\_CHANGE table as follows:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | End date of the NM\_MEMBERS\_ALL record (NM\_MEMBERS\_ALL.NM\_END\_DATE |
| DATUM\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_OF |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_TYPE | DISTANCE BREAK |
| OPERATION | CLOSED |
| OLD\_BEGIN\_MEASURE | NM\_SLK |
| OLD\_END\_MEASURE | NM\_END\_SLK |
| NEW\_BEGIN\_MEASURE | Null |
| NEW\_END\_MEASURE | Null |
| CHANGE\_START\_MEASURE | NM\_SLK |
| CHANGE\_END\_MEASURE | NM\_END\_SLK |
| MILEAGE\_CHANGE | NM\_SLK - NM\_END\_SLK |
| ROUTE\_ID | NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN |
| ROUTE\_UNIQUE | Nm3net.get\_ne\_unique(NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN) |
| ROUTE\_NAME | Nm3net.get\_ne\_descr(route\_ID) |

### Recalibrate Operation

When the operation is one of ‘B’ (Recalibrate) then a record is required and the process ‘New Datum Today‘ is processed next.

### New Datum Today

It is possible (although unlikely) that a new datum will be created and recalibrated on the same day. The NM\_ELEMENT\_HISTORY.NEH\_OPERATION will report ‘B’ so all recalibrate operations must be checked to see if they were also created on the same day.

First NM\_ELEMENT\_HISTORY will be checked to confirm this datum was not created with the same modified date as a split or merge or shift or replace or reverse operation. If it was, then the process ‘Recalibrate Datum’ can be processed next as this datum was not a new datum.

If it was not created by another operation, the NM\_ELEMENTS.NE\_START\_DATE of the datum must be checked to see if it equals the NM\_ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE of the recalibrate operation. If it does, then this datum was created in the same day as the recalibrate and should be processed by the ‘New Datum’ process. The New Datum process will use the recalibrated measures so the length change will be reported correctly.

### Recalibrate Datum

When a datum has been recalibrated, a record must be inserted into the XOR\_LENGTH\_CHANGE table as follows:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | Start date of the NM\_MEMBERS\_ALL record (NM\_MEMBERS\_ALL.NM\_START\_DATE |
| DATUM\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_OF |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_TYPE | DATUM |
| OPERATION | If OLD\_END\_MEASURE > NEW\_END\_MEASURE  Then ‘RECALIBRATE SHORTER’  Else if OLD\_END\_MEASURE < NEW\_END\_MEASURE  Then ‘RECALIBRATE LONGER’ |
| OLD\_BEGIN\_MEASURE | NM\_SLK + NEH\_PARAM\_1 |
| OLD\_END\_MEASURE | NM\_SLK + NEH\_OLD\_NE\_LENGTH |
| NEW\_BEGIN\_MEASURE | NM\_SLK + NEH\_PARAM\_1 |
| NEW\_END\_MEASURE | NM\_SLK + NEH\_PARAM\_1 + NEH\_PARAM\_2 |
| CHANGE\_START\_MEASURE | Smaller of NEW\_END\_MEASURE or OLD\_END\_MEASURE |
| CHANGE\_END\_MEASURE | Larger of NEW\_END\_MEASURE or OLD\_END\_MEASURE |
| MILEAGE\_CHANGE | OLD\_END\_MEASURE - NEW\_END\_MEASURE |
| ROUTE\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_IN |
| ROUTE\_UNIQUE | Nm3net.get\_ne\_unique(NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN) |
| ROUTE\_NAME | Nm3net.get\_ne\_descr(ROUTE\_ID) |

### Reclassify Operation

When the operation is one of ‘R’ (Reclassify) then multiple records are required and the process ‘Reclassify Datum’ is processed next.

### Reclassify Datum

When a datum has been reclassified so that it is transferred from one group to another, then two records must be inserted into the XOR\_LENGTH\_CHANGE table. A datum has been moved from one route to another, causing a length reduction in one and a length increase in the other.

The first record will be based on the current NM\_MEMBERS\_ALL record being processed:

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | Start date of the NM\_MEMBERS\_ALL record (NM\_MEMBERS\_ALL.NM\_START\_DATE |
| DATUM\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_OF |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_TYPE | DATUM |
| OPERATION | ADDED RECLASSIFY |
| OLD\_BEGIN\_MEASURE | null |
| OLD\_END\_MEASURE | null |
| NEW\_BEGIN\_MEASURE | NM\_SLK |
| NEW\_END\_MEASURE | NM\_END\_SLK |
| CHANGE\_START\_MEASURE | NM\_SLK |
| CHANGE\_END\_MEASURE | NM\_END\_SLK |
| MILEAGE\_CHANGE | NM\_END\_SLK - NM\_SLK |
| ROUTE\_ID | NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN |
| ROUTE \_UNIQUE | Nm3net.get\_ne\_unique(NM\_MEMBERS\_ALL. NM\_NE\_ID\_IN) |
| ROUTE \_NAME | Nm3net.get\_ne\_descr(ROUTE\_ID) |

The NM\_MEMBERS\_ALL record being removed will have an end date equal to the NM\_MEMBERS\_ALL record being added. A second record is required to record the reduction in length to the route this datum has been removed from. This will require information from both the NM\_MEMBERS\_ALL record being processed, the NM\_ELEMENT\_HISTORY table and the NM\_MEMBERS\_ALL table.

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | NM\_MEMBERS\_ALL.NM\_MODIFIED\_DATE |
| EFFECTIVE DATE | Start date of the NM\_MEMBERS\_ALL record (NM\_MEMBERS\_ALL.NM\_START\_DATE |
| DATUM\_ID | NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD) |
| DATUM\_TYPE | DATUM |
| OPERATION | CLOSED RECLASSIFY |
| OLD\_BEGIN\_MEASURE | SELECT NM\_SLK from nm\_members\_all where nm\_end\_date = ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE  and NM\_NE\_ID\_OF = NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| OLD\_END\_MEASURE | SELECT NM\_END\_SLK from nm\_members\_all where NM\_END\_DATE = ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE  And nm\_ne\_id\_of = NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| NEW\_BEGIN\_MEASURE | null |
| NEW\_END\_MEASURE | null |
| CHANGE\_START\_MEASURE | SELECT NM\_SLK from nm\_members\_all where nm\_end\_date = ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE  And nm\_ne\_id\_of = NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| CHANGE\_END\_MEASURE | SELECT NM\_END\_SLK from nm\_members\_all where nm\_end\_date = ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE  And nm\_ne\_id\_of = NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| MILEAGE\_CHANGE | OLD\_BEGIN\_MEASURE - OLD\_END\_MEASURE |
| ROUTE\_ID | SELECT NM\_NE\_ID\_IN from nm\_members\_all where nm\_end\_date = ELEMENT\_HISTORY.NEH\_EFFECTIVE\_DATE  And nm\_ne\_id\_of = NM\_ELEMENT\_HISTORY.NEH\_NE\_ID\_OLD |
| ROUTE\_UNIQUE | Nm3net.get\_ne\_unique(ROUTE \_ID) |
| ROUTE \_NAME | Nm3net.get\_ne\_descr(ROUTE \_ID) |

### Geometry Only Changes

Using XAA\_SPATIAL\_AUDIT table of data, relevant fields will be filled in to the XOR\_LENGTH\_CHANGE table.

|  |  |
| --- | --- |
| CHANGE\_ID | XOR\_LENG\_CHANGE\_SEQ.nextval |
| CHANGE\_DATE | XAA\_SPATIAL\_AUDIT.OP\_DATE |
| EFFECTIVE DATE | XAA\_SPATIAL\_AUDIT.EFF\_DATE |
| DATUM\_ID | NM\_MEMBERS\_ALL.NM\_NE\_ID\_OF |
| DATUM\_ UNIQUE | Nm3net.get\_ne\_unique (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_LENGTH | Nm3net.get\_ne\_length (NM\_MEMBERS\_ALL. NM\_NE\_ID\_OF) |
| DATUM\_TYPE | GROUP |
| OPERATION | GEOMETRY |
| OLD\_BEGIN\_MEASURE | Min(NM\_SLK ) |
| OLD\_END\_MEASURE | Max(NM\_END\_SLK) |

### Length Change Report

Using XOR\_LENGTH\_CHANGE table of data, the Length Change Report can be created. The report will need to specify a date range either coded into the logic of the report or as parameters of the report. The report will then retrieve all rows where the Change Date is within this range and attempt to group these operations where several operations result in a single effective operation.

To group the operations the report will be ordered by route, and milepoint. Where sequential operations exist they will be merged so that the report is easier to use. For example, if a route was made longer with the addition of several datum, these should be combined into a single line in the report listing an increase in length.

Some network changes, like a realignment, will require both the addition and removal of road network. Both will be reported separately.

### Cases Not Dealt with Explicitly

This section details some cases that might arise during normal processing but that aren’t mentioned explicitly above. The Length Change Report will cater for these cases.

* There may be a recalibrate and a subsequent merge or split on the same day. The result is that the member datum will have an end date but will be listed in the NM\_ELEMENT\_HISTORY table as both a recalibrate and a merge/split. Recalibrate operations are processed even if the NM\_MEMBERS\_ALL record has an End Date, and the merge or split will be ignored so the result will be correct.

# Documentation Requirements

Documentation is required to administer and run the interface. It needs to include detail on the configuration of the interface including the configuration of the interface metadata, and detail on how to execute the interface both manually and via a job.

# Assumptions

All asset information sent to the interface is based on a single asset record and asset records don’t need to be combined. If this is not correct, then a merge query will need to be employed and this will significantly increase the complexity required.

# Conclusion

This document is the result of a series of conversations between Bentley Systems, KYTC and AgileAssets with the objective of establishing an interface between the AgileAssets Maintenance System (including a PMS) and the Exor system KYTC uses to manage the road network and key asset data. From these discussions, Bentley Systems has created a software design that meets the agreed scope of this project and the requirements that need to be met in order for the project to be successful.

Using this design, Bentley Systems will create an estimate for the development and deployment of the software that details how the interface software will meet the requirements and an estimate of effort required to complete the project.

# Appendix I

Asset information stored by Exor that must be exchanged with AgileAssets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Asset Type | Attribute | EXOR Field | Agile Field Name | Notes |
| AL | AUXLANE | IIT\_CHR\_ATTRIB26 | Not in system |  |
|  | AUXLNWID | IIT\_NUM\_ATTRIB16 | Not in system |  |
|  | AUXSURF | IIT\_CHR\_ATTRIB37 | Not in system |  |
|  |  |  |  |  |
| FS | URBAREA | IIT\_CHR\_ATTRIB39 | PMS.Custom\_5 | Not used currently, FUNCT is still reported as PMS.Cur\_Pave\_MGMT\_Sections.Class\_7 |
|  | STATUS | IIT\_CHR\_ATTRIB27 | PMS.Cur\_Pave\_MGMT\_Sections.In\_Use |  |
|  | FC | IIT\_CHR\_ATTRIB28 | PMS.Custom\_7 | Not used currently, FUNCT is still reported as PMS.Cur\_Pave\_MGMT\_Sections.Class\_7 |
|  | NHS | IIT\_CHR\_ATTRIB45 | PMS.Cur\_Pave\_MGMT\_Sections.Class\_5 | Class variables are defined in separate tables, shown on excel sheets below |
|  |  |  |  |  |
| LN | LANEWID | IIT\_NUM\_ATTRIB16 | PMS.Cur\_Pave\_MGMT\_Sections.Sec\_Width | Currently lists the result of # Lanes x Ln Width |
|  | LANES | IIT\_NUM\_ATTRIB17 | PMS.Cur\_Pave\_MGMT\_Sections.Number\_of\_Lanes |  |
|  | LANESCRD | IIT\_NUM\_ATTRIB18 | Not in system |  |
|  | LANESNC | IIT\_NUM\_ATTRIB19 | Not in system |  |
|  |  |  |  |  |
| RA | SAFEINDX | IIT\_NUM\_ATTRIB19 | Not in system |  |
|  | SERVINDX | IIT\_NUM\_ATTRIB20 | Not in system |  |
|  | COMPINDX | IIT\_NUM\_ATTRIB21 | Not in system |  |
|  | PERCENTILE | IIT\_NUM\_ATTRIB24 | Not in system |  |
|  |  |  |  |  |
| RW | ROW\_WIDTH | IIT\_NUM\_ATTRIB16 | Not in system |  |
|  |  |  |  |  |
| SH | SHLDTYPE | IIT\_CHR\_ATTRIB26 | Not in system |  |
|  | SHLDWID | IIT\_NUM\_ATTRIB16 | Not in system |  |
|  |  |  |  |  |
| SL | SPEEDLIM | IIT\_NUM\_ATTRIB16 | Not in system |  |
|  | OONUMBER | IIT\_NUM\_ATTRIB17 | Not in system |  |
|  |  |  |  |  |
| SS | STHWYSYS | IIT\_CHR\_ATTRIB38 | PMS.Cur\_Pave\_MGMT\_Sections.Class\_3 |  |
|  |  |  |  |  |
| TF | LASTCNT | IIT\_NUM\_ATTRIB25 | PMS.Traffic.ADT |  |
|  | LASTCNTYR | IIT\_NUM\_ATTRIB22 | Not in system |  |
|  | ADTSINGLE | IIT\_NUM\_ATTRIB80 | Not in system |  |
|  | ADTCOMBO | IIT\_NUM\_ATTRIB81 | PMS.Traffic.ESAL |  |
|  | PCSINGOP | IIT\_NUM\_ATTRIB17 | Not in system |  |
|  | PCCOMBOP | IIT\_NUM\_ATTRIB18 | Not in system |  |
|  | PCSINGPK | IIT\_NUM\_ATTRIB23 | Not in system |  |
|  | PCCOMBPK | IIT\_NUM\_ATTRIB77 | Not in system |  |