# INCREMENTAL UPDATING USING THE GOTHIC VERSIONED OBJECT DATABASE WITH THE HYDROGRAPHIC S57 ENC AND SOTF SPATIAL OBJECT TRANSFER FORMATS

**Paul Hardy &**
**Peter Woodsford**
Laser-Scan Ltd., Cambridge, UK
paul@lsl.co.uk, peterw@lsl.co.uk

## ABSTRACT

Traditional relational databases and existing spatial transfer formats have been shown to be inadequate for creation, update and dissemination of the geospatial data so fundamental to modern life and commerce. This realisation has led to a shift to the Object paradigm for modelling geospatial data.

This paper overviews the versioning and incremental update capabilities of a commercial object-oriented geospatial database. It specifically highlights its capabilities for multi-user simultaneous write access to continuous map data and the importance of versioned objects for generation of minimal update messages. It uses as examples the real-world implementation of the hydrographic electronic navigation chart (S57 ENC) service, with its weekly updates to mariners.

It then outlines a new Spatial Object Transfer Format (SOTF), prototyped for the US National Imagery and Mapping Agency (NIMA). SOTF is specifically designed for the transfer of spatial object data, including provision of incremental updates, the preservation of added-value data and for user-defined areas-of-interest. SOTF uses XML, and has been proposed to the OpenGIS Consortium (OGC) as the direction for the future evolution of the Geography Markup Language (GML).

## 1 INTRODUCTION

### 1.1 Gothic

Laser-Scan Gothic is an object-oriented (O-O) spatial database and toolkit [1] with a family of layered applications, ranging from map and chart production [2] through store placement and remote-sensed imagery analysis to desktop bespoke solutions and active web mapping [3]. Gothic is designed specifically for handling very large volumes of continuous geographic object data, and for efficient multi-user scalability. A typical Gothic installation consists of a database server and several client workstations or application servers (Unix or Windows NT), sharing data across a local area network (e.g. Figure 1).

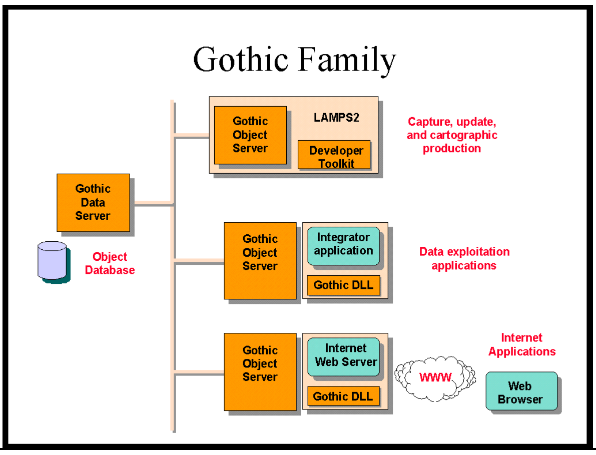


Figure 1: Gothic Family of Applications

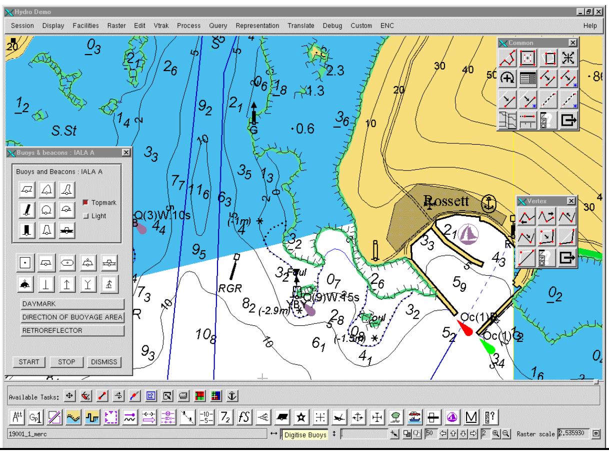


Fig. 2 – LAMPS2 Hydrographic application

Gothic was a clean implementation starting afresh in the early 90s, and hence does not have to retain any impeding baggage from previous generation file-based or feature-based systems. Gothic solutions have been installed at hundreds of sites world-wide, particularly in National Mapping Agencies (NMAs) and large commercial mapping organisations. The scope of the environment is very wide, and for instance, Gothic is used for facilities management of utilities in Korea. Most of the screen shot examples of this paper (see Figure 2) are taken from hydrographic production flowlines (data courtesy of UKHO, Crown Copyright).

### 1.2 Gothic Objects

The heart of the Gothic environment is the object database. The entities stored in the database are true O-O object instances, with properties and behaviours defined on the object classes in the database schema. The schema supports multiple inheritance between object classes, thus allowing a Gothic dataset to model the entities from the real world (roads, buildings, and rivers), in their family relationships, with their properties (attributes), behaviours (methods), and references (pointers).

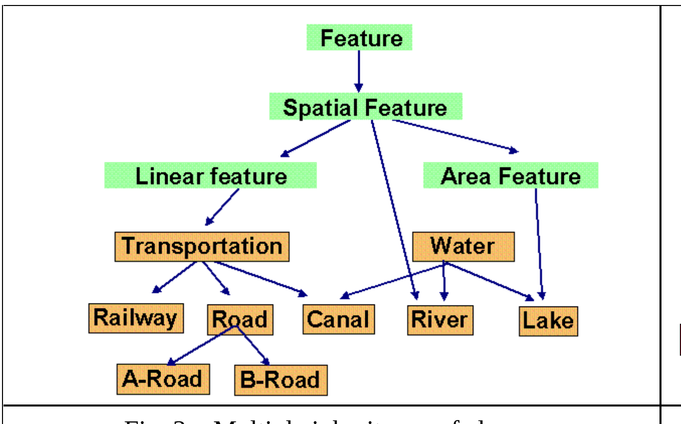


Fig. 3 – Multiple inheritance of classes

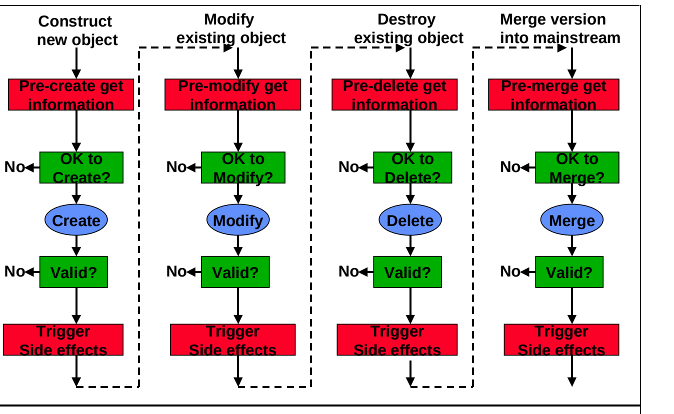


Figure 4: Gothic Object Lifecycle Methods

An important aspect of Gothic objects is that there is a defined object lifecycle (see Figure 4), in which the database will automatically trigger methods at important stages in an object's existence. These 'reflex methods' are key to many important capabilities of the Gothic environment, such as enforcing integrity, or maintaining topological structure.

## 2 GOTHIC VERSIONING

### 2.1 Dataset Versions

A Gothic database typically contains several independent datasets, each of which is maintained in a version tree structure, with only delta change information recorded rather than complete copies. Multi-user update is supported, with each update process accessing its own logical copy of the complete dataset, without the overhead of providing a physical copy to each process (Figure 5). Permission of the Gothic Database Access Manager (DAM) is only needed at start and end of an update session. Thereafter each session has its own stable version and is not affected by the actions of other uses. This is a contrast to a legacy relational database architecture, which relies on a central database server through which all edits have to be funnelled, and which handles locking. Transaction management in Gothic is based on 'long transactions', defined by 'segments' (see Figure 6).

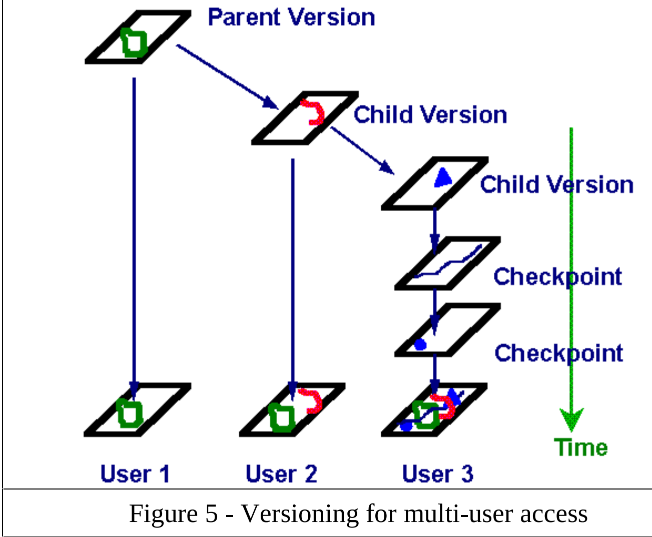


Figure 5 - Versioning for multi-user access

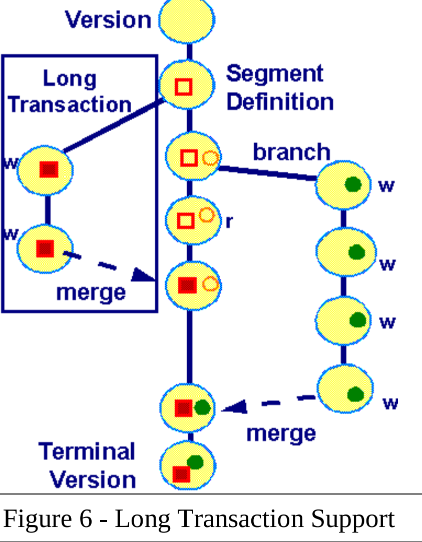


Figure 6 - Long Transaction Support

## 2.2 Long Transactions

Segments are used to identify objects for which exclusive write access has been granted to a particular user. Segments may be a spatial extent (e.g. a rectangle) or a set of object classes or a combination of these. Overlap of segments is allowed (with a warning) for ‘what if’ scenarios in which different alternative ideas of reality need to be modelled.

Within the long transaction, any of the Gothic editing or processing tools can be used to create new objects, modify existing objects, or delete objects. Validation methods are used to assure local data integrity.

Merge methods are used to carry out the final commit at the end of a long transaction, and assure global data integrity during the merge

## 2.3 Versioning of metadata

As well as the feature objects and underlying topology objects themselves, both the schema and the in-built spatial index are also versioned.

## 2.4 Version management

If all created versions were allowed to accumulate indefinitely, then this would adversely affect performance, and be a data management headache. A Gothic database management tool called MANAGE is provided which has facilities for version compression to remove versions no longer needed.

MANAGE is also used for backup and restore of datasets or complete database. Dataset backup can be full (all versions), or incremental (just versions created since last backup).

A diagnostic and repair utility OBDBTOOL is also provided, and this can be used for analysis of the version tree structure, and for less common management operations, such as cloning datasets.

## 3 VERSIONING AND INCREMENTAL UPDATES FOR ENC PRODUCTION

The Gothic-based S57 flowline has been under development at UKHO since 1996. It is now in production use, for both complete ENC Cells and weekly updates. It is also in use at NOAA and elsewhere, and is described more fully in the paper by Hardy (1998) on “S57 ECDIS Data Production and Update using an Object-Oriented Spatial Database” [5].

## 3.1 Versioning as a basis for Incremental Update

Because a version holds only the changes relative to the parent version, version comparison (differencing) and the generation of incremental updates is a cheap and simple operation. The cumulative effect of the delta changes can be summarised in the format required, after the validation stage in merging has been successfully accomplished. The S57 exporter software then tags the version corresponding to this Update stage as a baseline Version to be used as a starting point for the subsequent Update stage.

The Gothic Object database uses unique system Object-ID's to manage objects. Although these are unique across a Gothic database, they are not however unique across all production agencies as required by S57 Feature-Object ID's (FOIDS). For ENC updates two different unique identifiers are used:

- An internal record ID that is assigned to every node, edge, feature and collection object, which can be referred to during, updates within a given cell. These are only unique to a cell and are generated by incrementing an integer starting at one. The Laser-Scan ENC exporter allocates these whenever a new node, edge, feature and collection object is created. This ID can take integer values from 1 to 2**32-2.
- A world-wide feature identifier for each feature object. This consists of a 32-bit "Feature Identification Number" (FIDN) and a 16-bit "Feature Identification Subdivision" (FIDS). Together these offer a way of uniquely referring to feature objects across all cells within a given producing agency. The producing agency code (16-bit) is taken together with FIDN and FIDS to give a 64-bit unique code. Laser-Scan software does not set this code which is held as three separate attributes on each real-world object. UKHO software (for example) allocates these unique codes using a combination of user-ID and the current date and time to the nearest centisecond.

## 3.2 Specific Issues for ENC

Versioning is traditionally used in support of ‘Long Transactions’. The updating of a database holding ENC data, and the generation of the resultant Update messages, represents a ‘special case’ in the following respects:

- Duration/Frequency of transactions. Updates to ENC databases are typically relatively small and short, although, of course, some can be complex and therefore long. In addition, the number of update transactions can be quite significant, and large numbers of ‘checkpoints’ can be created in the interests of data security and the ability to backtrack. In consequence, complex version trees can arise. This is not in itself a problem providing the system is equipped with utility functions to manage and tidy up such trees. In particular a utility to compress non-terminal versions (such as the checkpoints in the figure above) is needed.
- Preservation of Baseline Versions. As described above, Baseline Versions have to be retained as required to provide the baseline for the differencing operations needed to support the generation of Update messages. It is essential that Baseline Versions are tagged and are preserved in all version-tree tidying utilities, or the incremental process breaks down.

It is worth noting that even the shortest spatial modification is generally too long to be handled as a single transaction in a relational database, but fits well with the ‘long transaction’ model of versioned object databases.

## 3.3 Requirements for Generating ‘Thin’ Update Messages

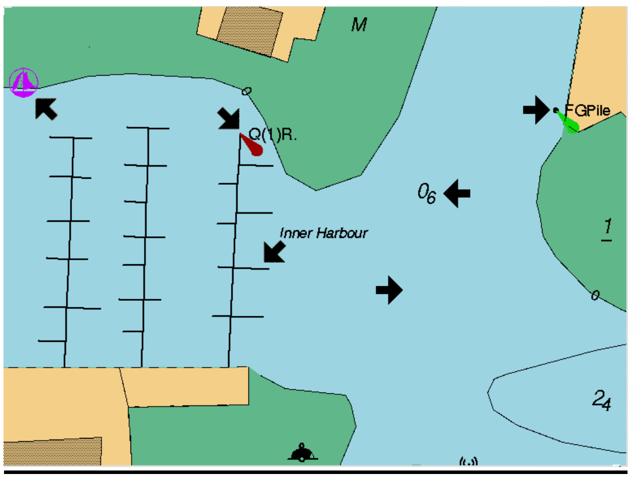
In the Gothic Object data model, there is a hierarchy:

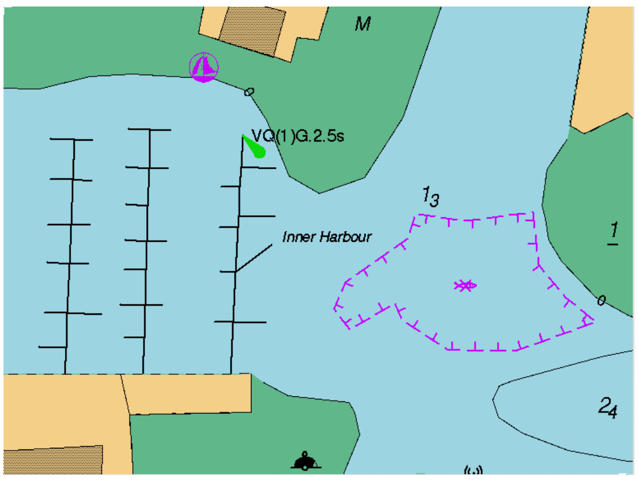
- Real World Objects (RWO’s), which model the real world.
- Topological Primitives (links, nodes, faces)

RWO’s can be point, line or area objects or collection objects. This hierarchy corresponds closely to the S57 concepts of ‘Feature Objects’ and ‘Spatial Objects’.

The ENC Flowline manages RWO’s corresponding to the S57 data catalogue, but the operations involved in updating result in manipulations at the topological primitive level. The naïve approach to implementing update operations would be to effect complete removals and replacements at the RWO level. When translated down to the topological primitive level, this results in excessively voluminous update messages. In order to minimise the volume of update messages generated when updating in a particular segment, or operational area, it is necessary to ‘insulate’ the segment by generating transient nodes at all intersections with the segment boundary. This provides freedom of action to optimise the effects of updates within the segment. The transient nodes, except any altered by the updates, are suppressed within the merge process, after all validations have been successfully completed.

This technique minimises the volume of update messages, except for cases that fall across cell boundaries (a small minority of cases).





| Before | After |
|---|---|

### 3.4 Example of ENC Incremental Update

The two figures above the ‘Before’ and ‘After’ displays from the database, and Appendix A contains the textual form of the Update messages generated by the resultant Incremental Update export. The comments of the form ‘*** words’ are inserted by way of commentary. This form of printout is used in development to check that messages are optimised.

In summary the Updates are:

- Move of mooring facility
- Modification to shoreline construction
- Change of sounding
- Change of attributes of lights
- Creation of new restricted area
- Deletion of (coincided) lights and pile point.

## 4 SPATIAL OBJECT TRANSFER FORMAT (SOTF)

### 4.1 Why SOTF ?

Many organisations, such as National Mapping Agencies (NMAs) whose business is the management and delivery of geospatial information are shifting their data holdings to object oriented data models residing in very large continuous datasets. A Spatial Object Transfer Format (SOTF) has been designed and prototyped to address the issues arising in transferring such data to users. It also addresses fundamental limitations in the current generation of transfer formats (inability to support incremental update, reliance on tiling, failure to preserve value-added data). SOTF is described more fully in a Laser-Scan white paper [5] and in Woodsford [6].

Despite progress in establishing interoperability standards (such as the OpenGIS interface), there remains a requirement for a standard transfer format for both archive purposes and for deployment in situations where continuous high bandwidth 'connection-by-wire' solutions are inappropriate. SOTF addresses these two requirements in a vendor-neutral manner based on mainstream IT technology and emerging standards.

### 4.2 An Introduction to SOTF.

SOTF is based on an XML 1.0 encoding for geospatial information. It builds on the GML (Geographic Markup Language) document from the OpenGIS consortium. However, XML based technologies are evolving rapidly, especially in the area used by SOTF, where there is a need to describe both data and schema. Consequently Laser-Scan is actively tracking related XML developments and expects to issues revisions to the SOTF specification accordingly.

SOTF allows for the transfer of a model of the real world. This model is constructed from features and their related geospatial information in a rich feature model based on emerging standards from the GIS community as well as established software engineering concepts, primarily those related to object orientation. Every feature has an identifier that is persistent. Thus if the features from each of two SOTF datasets (separated in time) are to be recognised as the same, they must have the same feature identifier. This is true even if the feature has been modified in the intervening period. This provides the basis of many of the new operations offered by SOTF. SOTF does not dictate the form of feature identifier a data producer must use, it merely requires that it can be encoded in a string. A data consumer can only rely on the equivalence of identifiers.

SOTF is designed to provide a means to transfer or archive highly structured geospatial information, with as little ambiguity as possible. It is not designed to capture all the elements of an object-oriented model; most notably it does not handle methods and their associated behaviour.

SOTF makes a distinction between features (with their associated feature types) and properties (with their associated data-types). Essentially a feature type is a named set of property types where a property type is a {name, data-type} pair and a property is the corresponding {name, data-type, value} triple. SOTF supports basic data-types (boolean, integer, real, string) that can be encoded as parsed character data and more complex data-types (geometry) that require XML encodings of their own. SOTF also allows the definition of role types, which allow SOTF to model binary, lightweight feature relationships as defined in the OpenGIS Consortium's Abstract Specification. Features may have zero or more geometric properties distinguished by name, just as they may have zero or more properties of any data-type.

### 4.3 SOTF Example feature

- Feature Identifier
- Feature type
- Dependencies
- Roles and relationship information
- Attribution
- Geometry property

```
<Feature fid="ID:139.677.10355552" featureType="Road (line)">
      <depend>
            <dependValue idref="SID:139.677.10355552"/>
      </depend>
      <role name="topoFeature">
            <featureValue idref="SID:139.677.10355552"/>
      </role>
      <role name="Over"/>
      <role name="Under"/>
      <Property value="ID:139.677.10355552" name="SOTF_identifier" type="string"/>
      <Property value="28" name="Existence Category" type="integer"/>
      <Property value="AP030" name="FACC Code" type="string"/>
      <Property value="cul transl 179" name="feature id" type="string"/>
      <Property name="geometry" type="geometry">
            <LString SRS_Name="srs:WGS.2084">
                  <CList dimension="2">
                        -76.189102,36.925549
                        -76.188782,36.926933
                        -76.188538,36.928013
                  </CList>
            </LString>
      </Property>
</Feature>
```

### 4.4 SOTF ’area-of-interest’, or AOI

Each SOTF dataset includes a definition of the AOI upon which it is based. The feature data within the AOI is guaranteed to be complete and correct. This includes the handling of feature relationships and the concepts built upon them, for example topology. An AOI allows a data producer to maintain data over a large extent but only return data for the AOI specified by the data consumer. SOTF achieves this without the need to introduce artificial constructs that break up features. The most common approach used by traditional transfer formats is to introduce a tiling scheme. Instead SOTF relies on having features implement the concepts of ’extent’ and ’dependency’:
- Every feature in a data store must be able to describe its spatial extent, typically as a Minimum Bounding Rectangle, or MBR.
- Every feature in a data store must be able to define the set of features upon which it is dependent. A feature is dependent on another feature if it cannot meaningfully exist in its absence, such as a line feature object being dependent on an underlying link object.

An SOTF dataset must include all the features whose extents overlap or are contained by the associated AOI *and* all the features upon which they are [recursively] dependent.

Users only receive bulk data and incremental update messages for their registered AOI’s. An associated advantage is that SOTF AOI’s can be combined. This allows a data producer to pre-define a set of AOI's and generate the corresponding set of SOTF datasets. Requests by data consumers that specify their own AOI can be answered by supplying one or more pre-generated SOTF datasets, which are then combined by the data consumer. This is particularly important when a data producer expects to deliver through media such as CD-ROM. Under such circumstances, a data producer may wish to stockpile data in a form ready for delivery to data consumers.

Because an SOTF dataset may include features outside its associated AOI, a number of SOTF datasets may include the same feature multiple times. However, the persistence of feature identifiers ensures that such situations can be identified and handled appropriately.

## 4.5 SOTF supports incremental update.

Using this, a data consumer that has entered an on-going relationship with a data producer can receive small, incremental updates detailing just the changes as the producer makes new data available. SOTF tags the features within a dataset as being 'new', 'modified' or 'deleted' relative to some previous state. The feature represents the lowest level of granularity that SOTF supports. A data consumer receiving an initial SOTF transfer would discover that all the features have been tagged as 'new'.

SOTF is defined to support three specific types of change within the incremental update. The incremental update data will be defined within wrappers defining the scope of each type of transaction.

- *Deleted feature* - The incremental transfer will include an abbreviated form of the feature just containing the feature's unique identifier within the source dataset. On receiving a deleted feature the end application will read the unique identifier and find and remove the target object with that given identifier.
- *Feature creation* - The incremental transfer can include complete new features. The unique identifier for each new feature must not match that used for any other features in the target dataset.
- *Feature update* – Modified features will be matched in the target dataset by their unique identifier, and contain the set of attributes with their updated values. Values for attributes not included in the update data are assumed unchanged. This is specifically required to support value-added target datasets.

## 4.6 Examples of each type of transaction (modify, create, delete)

### 4.6.1 XML for Modified Road Feature

```
<featureModified>
        <Feature fid="SID:139.581.10355485" featureType="topoCurve2D">
                <depend>
                        <dependValue idref="ID:56.1416.9389490" />
                </depend>
                <role name="edgePrimitives">
                        <featureValue idref="FID:56.1416.9389490" />
                </role>
                <role name="userFeature">
                        <featureValue idref="ID:139.581.10355485" />
                </role>
        </Feature>
</featureModified>

<featureModified>
        <Feature fid="ID:139.581.10355485" featureType="Road (line)">
                <depend>
                        <dependValue idref="SID:139.581.10355485" />
                </depend>
                <role name="topoFeature">
                        <featureValue idref="SID:139.581.10355485" />
                </role>
                <role name="Over" />  <role name="Under" />
                <Property value="AP030" name="FACC Code" type="string" />
                <Property value="28" name="Existence Category" type="integer" />
                <Property value="cul transl 131" name="feature id" type="string" />
                <Property value="ID:139.581.10355485" name="SOTF_identifier" type="string" />
                <Property name="geometry" type="geometry">
                        <LString SRS_Name="srs:WGS.2084">
                                <CList dimension="2">-76.191696,36.924557 -76.191803,36.924088
                                -76.191931,36.923556 -76.19166,36.922779 -76.191712,36.922142
                                -76.191901,36.921814 -76.192263,36.921453 -76.19247,36.921212
                                -76.192479,36.921214 -76.192535,36.920979</CList>
                        </LString>
                </Property>
        </Feature>
</featureModified>
```

### 4.6.2 XML for Created Edge and Road Feature

```
<featureCreated>
        <Feature fid="ID:56.1648.9401318" featureType="edge">
                <depend>
                        <dependValue idref="ID:54.1652.9402744" />
                        <dependValue idref="ID:54.1259.9401141" />
                </depend>
                <Property name="geometry" type="curve">
                        <LString SRS_Name="srs:WGS.2084">
                                <CList dimension="2">-76.189667,36.928181
                                -76.189928,36.927066</CList>
                        </LString>
                </Property>
                <role name="nodes">
                        <featureValue idref="ID:54.1652.9402744" />
                        <featureValue idref="ID:54.1259.9401141" />
                </role>
                <role name="direction">
                        <featureValue idref="FID:56.1648.9401318" />
                        <featureValue idref="BID:56.1648.9401318" />
                </role>
        </Feature>
</featureCreated>
…
<featureCreated>
        <Feature fid="ID:139.1655.10095793" featureType="Road (line)">
                <depend>
                        <dependValue idref="SID:139.1655.10095793" />
                </depend>
                <role name="topoFeature">
                        <featureValue idref="SID:139.1655.10095793" />
                </role>
                <role name="Over" />  <role name="Under" />
                <Property name="geometry" type="geometry">
                        <LString SRS_Name="srs:WGS.2084">
                                <CList dimension="2">-76.186241,36.92482
                                -76.184158,36.924419</CList>
                        </LString>
                </Property>
        </Feature>
</featureCreated>
```

### 4.6.3 XML for Deleted Feature

```
<featureDeleted featureid="ID:54.1419.9397112" />
```

## 4.7 Preservation of User Value-Added Information

There are many situations where the schema defined by a data producer is inadequate for a data consumer. Typically, a data consumer may wish to augment the supplied data with data of their own. For example, a data consumer may wish to add a boolean property called 'visited' to a 'city' feature type. SOTF defines the rules to determine if the schema from a data provider is 'consistent' with the schema currently in use by a data consumer. This allows a consumer to extend the schema to add value. It also allows the schema to be enhanced at the producer data store and have the changes applied at the consumer data store. The value-added data provided by a data consumer is preserved under incremental update from the source data producer. Value-adding is made possible by the SOTF requirement that the schema is described explicitly in an SOTF dataset.

The mechanisms that SOTF introduces to handle incremental update, AOI and combination integrate a number of concepts. Consequently, the SOTF specification is more than just the encoding scheme used to transfer the feature data itself, it includes the logic by which SOTF exporters and importers operate.

### 4.8 SOTF - Current Status and Way Ahead

Laser-Scan has completed an initial contract with NIMA (NMA201-98-3-0022) to research the requirements for an SOTF, to produce a High Level Design and deliver a working prototype based on a subset of the Digital Nautical Chart (DNC) profile of the Vector Product Format (VPF) standard. The research phase combined an analysis of existing transfer standards across the geospatial community with a requirements analysis. The prototype covers the essential elements of the SOTF requirements to the proof-of-concept level. Appendix B contains a partial DTD (Data Type Description) for the initial version of SOTF. Not included are the handling of spatial reference systems (datums and projections), or of metadata definitions.

The current SOTF design was presented to a meeting of the OpenGIS Consortium on April 4th, 2000 in San Bernardino, CA, as the basis of a way for forward for the Geography Markup Language (GML) standard. GML is intended for both bulk delivery and for transfer at the feature level across the Web.

## 5 CONCLUSIONS

- The work described and the experience gained with S57 and SOTF reinforces the view that managing geospatial information as objects with persistent unique identifiers is the key to delivering an Update service that supports complex data models and relationships and preserves value-added information at the receiver (user) end. The technique of Versioning, as well as providing multi-access update control, provides powerful tools (versioning differencing, active object lifecycle) for the efficient generation of Update messages. The SOTF project has shown that complex object schema and data can be expressed and transferred in XML. It has demonstrated Incremental Update with preservation of user-defined added-value data, and introduced the concept of user-defined areas-of-interest as an alternative to rigid, pre-defined tiling structures.
- The concepts established in SOTF potentially provide a neutral XML encoding for bulk and incremental transfers in issuer/receiver (supplier/user) scenarios across non-live connections, for transaction services between interoperating processes/data stores and across the Web, and for archival purposes. The potential exists for a unifying standard across these diverse requirements.

## ACKNOWLEDGEMENTS

The SOTF prototype development has been carried out as a collaborative project between the US National Imagery and Mapping Agency (NIMA) and Laser-Scan, (Sterling Va. USA and Cambridge UK).

The S57 ENC production flowline has been implemented jointly by the United Kingdom Hydrographic Office (UKHO) and Laser-Scan.

## REFERENCES

[1] Laser-Scan Gothic Product Overview, July 2000, http://www.laser-scan.com/products/

[2] Laser-Scan Gothic LAMPS2 Overview, July 2000, http://www.laser-scan.com/products/lamps2.htm

[3] Laser-Scan Gothic Integrator: Java Edition, July 2000, http://www.laser-scan.com/products/webbrowser.htm

[4] Hardy, P. G. ‘S57 ECDIS Data Production and Update using an Object-Oriented Spatial Database’, International ECDIS Conference, Singapore, October 1998.

[5] Spatial Object Transfer Format (SOTF) – a White Paper, April 2000, http://www.laser-scan.com/papers/sotf-apr14.html

[6] Woodsford, P. A. ‘A Prototype Spatial Object Transfer Format (SOTF)’, 6th EC-GI&GIS Workshop, Lyon, France, June 2000.

## APPENDIX A – SAMPLE S57 ENC UPDATES

```
0001: 0
DSID: RCNM DS RCID 1 EXPP 2 INTU 3 DSNM
GB300003.001 EDTN 1 UPDN 1 UADT  <NULL>
ISDT 19961201 STED 3.000000 PRSP 1 PSDN
<NULL> PRED 1.0 PROF 2 AGEN GB COMT
<NULL>

DSSI: DSTR 2 AALL 1 NALL 1 NOMR  <NULL>
NOCR  <NULL> NOGR 9 NOLR  <NULL> NOIN 5
NOCN 7 NOED 9 NOFA  <NULL>

*** New sounding array
0001: 1
VRID: RCNM VI RCID 187 RVER 1 RUIN 1
SG3D: -1625184.000000,3044650.000000, 17
      -1625098.000000,3044407.000000, 13
      -1625175.000000,3044483.000000, 24
      -1625110.000000,3044486.000000, -10
      -1625671.000000,3044948.000000, 14
      -1625669.000000,3045067.000000, 41
      -1625554.000000,3044959.000000, 26
      -1625502.000000,3045090.000000, 47
      -1625418.000000,3044970.000000, 9
      -1625750.000000,3045374.000000, 61

*** Delete sounding array from SOUNDG
0001: 2
VRID: RCNM VI RCID 176 RVER 2 RUIN 2

*** Deleted isolated node to move HRBFAC
0001: 3
VRID: RCNM VI RCID 47 RVER 2 RUIN 2

*** New isolated node for moved HRBFAC
0001: 4
VRID: RCNM VI RCID 186 RVER 1 RUIN 1
SG2D: -1625051.000000,3044307.000000

*** Isolated node for PILPNT deleted
0001: 5
VRID: RCNM VI RCID 62 RVER 2 RUIN 2

*** Start/end node for edge of RESARE
0001: 6
VRID: RCNM VC RCID 1178 RVER 1 RUIN 1
SG2D: -1625144.000000,3044477.000000

*** Deleted VC for edge: 1259
0001: 7
VRID: RCNM VC RCID 980 RVER 2 RUIN 2

*** New node for replacement edge in
SLCONS
0001: 8
VRID: RCNM VC RCID 1179 RVER 1 RUIN 1
SG2D: -1625125.000000,3044314.000000

*** Deleted VC for edge 1260
0001: 9
VRID: RCNM VC RCID 979 RVER 2 RUIN 2

*** New start node for SLCONS edge 1456
0001: 10
VRID: RCNM VC RCID 1180 RVER 1 RUIN 1
SG2D: -1625126.000000,3044321.000000

*** Deleted VC for edge 1260
0001: 11
VRID: RCNM VC RCID 981 RVER 2 RUIN 2

*** New end node for SLCONS edge 1456
0001: 12
VRID: RCNM VC RCID 1181 RVER 1 RUIN 1
SG2D: -1625116.000000,3044336.000000

*** New edge for RESARE, node: 1178 (start
and end)
0001: 13
VRID: RCNM VE RCID 1454 RVER 1 RUIN 1
VRPT: NAME (B(24)) : 0x789a04 ORNT 255
USAG 255 TOPI 1 MASK 255
      NAME (B(24)) : 0x789a04 ORNT 255
USAG 255 TOPI 2 MASK 255
SG2D: -1625135.000000,3044464.000000
      -1625123.000000,3044452.000000
      -1625104.000000,3044450.000000
      -1625107.000000,3044420.000000
      -1625104.000000,3044399.000000
      -1625113.000000,3044396.000000
      -1625129.000000,3044368.000000
      -1625139.000000,3044363.000000
      -1625147.000000,3044370.000000
      -1625137.000000,3044390.000000
      -1625143.000000,3044393.000000
      -1625154.000000,3044413.000000
      -1625154.000000,3044442.000000

*** Deleted edge for SLCONS (884)
0001: 14
VRID: RCNM VE RCID 1262 RVER 2 RUIN 2

*** New edge for SLCONS (884)
0001: 15
VRID: RCNM VE RCID 1455 RVER 1 RUIN 1
VRPT: NAME (B(24)) : 0x78d003 ORNT 255
USAG 255 TOPI 1 MASK 255
      NAME (B(24)) : 0x789c04 ORNT 255
USAG 255 TOPI 2 MASK 255

*** Deleted edge for SLCONS (883)
0001: 16
VRID: RCNM VE RCID 1259 RVER 2 RUIN 2

** Replacement edge for SLCONS
0001: 17
VRID: RCNM VE RCID 1456 RVER 1 RUIN 1
VRPT: NAME (B(24)) : 0x789b04 ORNT 255
USAG 255 TOPI 1 MASK 255
      NAME (B(24)) : 0x789c04 ORNT 255
USAG 255 TOPI 2 MASK 255

*** Deleted edge for SLCONS (883)
0001: 18
VRID: RCNM VE RCID 1260 RVER 2 RUIN 2

*** Replacement edge for SLCONS, nodes:
start: 1180, end: 1181
0001: 19
VRID: RCNM VE RCID 1457 RVER 1 RUIN 1
VRPT: NAME (B(24)) : 0x789c04 ORNT 255
USAG 255 TOPI 1 MASK 255
      NAME (B(24)) : 0x789d04 ORNT 255
USAG 255 TOPI 2 MASK 255
```

```
***
0001: 20
VRID: RCNM VE RCID 1263 RVER 2 RUIN 2

*** New edge for modified SLCONS (884)
0001: 21
VRID: RCNM VE RCID 1458 RVER 1 RUIN 1
VRPT: NAME (B(24)) : 0x789c04 ORNT 255
USAG 255 TOPI 1 MASK 255
      NAME (B(24)) : 0x78d603 ORNT 255
USAG 255 TOPI 2 MASK 255

*** Deletion of sounding feature
0001: 22
FRID: RCNM FE RCID 1062 PRIM 1 GRUP 255
OBJL SOUNDG RVER 2 RUIN 2

*** Creation of new sounding feature, VI =
187
0001: 23
FRID: RCNM FE RCID 1073 PRIM 1 GRUP 255
OBJL SOUNDG RVER 1 RUIN 1
FOID: AGEN GB FIDN  <NULL>
FIDS  <NULL>
ATTF: QUASOU=<1>
FSPT: NAME (B(24)) : 0x6ebb00 ORNT 255
USAG 255 MASK 255

*** Moved HRBFAC, new VI: 186
0001: 24
FRID: RCNM FE RCID 535 PRIM 1 GRUP 2 OBJL
HRBFAC RVER 2 RUIN 3
FOID: AGEN GB FIDN 1081 FIDS 1
FSPC: FSUI 3 FSIX 1 NSPT 1
FSPT: NAME (B(24)) : 0x6eba00 ORNT 255
USAG 255 MASK 255

*** Change LIGHTS - attributes only
0001: 25
FRID: RCNM FE RCID 572 PRIM 1 GRUP 2 OBJL
LIGHTS RVER 2 RUIN 3
FOID: AGEN GB FIDN 1199 FIDS 1
ATTF: COLOUR=<4> LITCHR=<5> SIGPER=<2.5>

*** Created RESARE, new edge: 1454
0001: 26
FRID: RCNM FE RCID 1072 PRIM 3 GRUP 255
OBJL RESARE RVER 1 RUIN 1
FOID: AGEN GB FIDN 175527920 FIDS 477
ATTF: RESTRN=<3>
FSPT: NAME (B(24)) : 0x82ae05 ORNT 2 USAG
1 MASK 255

*** Modification to SLCONS; replace
pointers to edges with: 1456, 1457
0001: 27
FRID: RCNM FE RCID 883 PRIM 2 GRUP 2 OBJL
SLCONS RVER 2 RUIN 3
FOID: AGEN GB FIDN 1425 FIDS 1
FSPC: FSUI 3 FSIX 1 NSPT 2
FSPT: NAME (B(24)) : 0x82b005 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82b105 ORNT 1 USAG
255 MASK 255

*** Modification to SLCONS; replace
pointers to edges with: 1261, 1455, 1458,
1264, 1265, 1266, 1267
0001: 28
FRID: RCNM FE RCID 884 PRIM 2 GRUP 2 OBJL
SLCONS RVER 2 RUIN 3
FOID: AGEN GB FIDN 1426 FIDS 1
FSPC: FSUI 3 FSIX 1 NSPT 7
FSPT: NAME (B(24)) : 0x82ed04 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82af05 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82b205 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82f004 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82f104 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82f204 ORNT 1 USAG
255 MASK 255
      NAME (B(24)) : 0x82f304 ORNT 1 USAG
255 MASK 255

*** Delete LIGHTS
0001: 29
FRID: RCNM FE RCID 574 PRIM 1 GRUP 2 OBJL
LIGHTS RVER 2 RUIN 2

*** Delete PILPNT
0001: 30
FRID: RCNM FE RCID 716 PRIM 1 GRUP 2 OBJL
PILPNT RVER 2 RUIN 2
```

## APPENDIX B – PARTIAL SOTF DTD

```
<?xml version = "1.0" ?>
<!ELEMENT SOTF (metadata, schema, featureCollection*, featureUpdate*)>

<!ELEMENT featureCollection (Feature*)>
<!ATTLIST featureCollection
      name CDATA   #REQUIRED>
<!ENTITY % Geometry  "(GeometryCollection | Point | LString | Polygon | MultiPoint |
              MultiLine | MultiPolygon)" >
<!ENTITY % LRing  "CList+" >
<!ELEMENT Description  (#PCDATA) >
<!ELEMENT Envelope  (Polygon) >
<!ENTITY % AttributeValue "(role|Property)">
<!ELEMENT Feature  (depend*,(%AttributeValue;)*,Feature*) >
<!ELEMENT depend (dependValue)*>
<!ELEMENT dependValue EMPTY>
<!ATTLIST dependValue
      idref CDATA  #REQUIRED>
<!ELEMENT FeatureCollection  (SpatialReferenceSystem+,Property*,Envelope?,Feature*) >
<!ELEMENT Geographic  (Datum,%AngularUnit;) >
<!ELEMENT GeometryCollection  (%Geometry;)+ >
<!ELEMENT LRing  (CList) >
<!ELEMENT LString  (CList) >
<!ELEMENT Link   EMPTY  >
<!ELEMENT MultiLine  (LString+) >
<!ELEMENT MultiPoint  (Point+) >
<!ELEMENT MultiPolygon  (Polygon+) >
<!ELEMENT Name  (#PCDATA) >
<!ELEMENT Point (CList) >
<!ELEMENT Polygon  (CList+) >
<!ELEMENT Property   (EMPTY|%Geometry;)*  >
<!ELEMENT SpatialReferenceSystem  %CS; >
<!ATTLIST CList
              dimension  CDATA    #REQUIRED  >
<!ATTLIST Feature
              fid  ID    #IMPLIED
              featureType  CDATA    #IMPLIED
              name  CDATA    #IMPLIED  >
<!ATTLIST LString
              SRS_Name  CDATA    #REQUIRED  >
<!ATTLIST MultiLine
              numLineStrings  CDATA    #IMPLIED
              SRS_Name  CDATA    #REQUIRED
              GeometryID  CDATA    #REQUIRED  >
<!ATTLIST MultiPoint
              numPoints  CDATA    #IMPLIED
              SRS_Name  CDATA    #REQUIRED
              GeometryID  CDATA    #REQUIRED  >
<!ATTLIST MultiPolygon
              numPolygons  CDATA    #IMPLIED
              SRS_Name  CDATA    #REQUIRED
              GeometryID  CDATA    #REQUIRED  >
<!ATTLIST Parameter
              Name  CDATA    #IMPLIED
              Value  CDATA    #IMPLIED  >
<!ATTLIST Point
              SRS_Name CDATA     #REQUIRED>
<!ATTLIST Polygon
              SRS_Name  CDATA    #REQUIRED
              numRings  CDATA    #IMPLIED  >
<!ATTLIST Property
              value  CDATA    #IMPLIED
              name  CDATA    #REQUIRED
              type  CDATA    #REQUIRED  >
<!ATTLIST SpatialReferenceSystem
              SRS_Name  ID    #REQUIRED  >
```

```
<!ENTITY % SchemaEntry "(themeType|relationshipType|featureType)">
<!ELEMENT schema (%SchemaEntry;)*>
<!ELEMENT themeType (superTheme)*>
<!ELEMENT superTheme EMPTY>
<!ELEMENT relationshipType (Description,(roleDef,roleDef)+)>
<!ELEMENT roleDef EMPTY>
<!ELEMENT relationshipRoles (roleUsed,roleUsed)+>
<!ELEMENT roleUsed EMPTY>
<!ENTITY % AttributeTypeEntry "(roleType|propertyType)">
<!ELEMENT featureType (theme*,superType*,geometry*,(%AttributeTypeEntry;)*)>
<!ELEMENT theme EMPTY>
<!ELEMENT superType EMPTY>
<!ELEMENT geometry EMPTY>
<!ELEMENT roleType EMPTY>
<!ELEMENT propertyType EMPTY>
<!ATTLIST themeType
      name   CDATA  #REQUIRED>
<!ATTLIST superTheme
      idref IDREF  #REQUIRED>
<!ATTLIST relationshipType
      name   ID     #REQUIRED
      degree  CDATA#REQUIRED>
<!ATTLIST roleDef
      name CDATA   #REQUIRED
      ordered CDATA#REQUIRED
      cardinality CDATA #REQUIRED>
<!ATTLIST roleUsed
      name CDATA   #REQUIRED>
<!ATTLIST featureType
      name CDATA   #REQUIRED>
<!ATTLIST propertyType
      name CDATA   #REQUIRED
      type CDATA   #REQUIRED
      multivalue CDATA    #REQUIRED>
<!ATTLIST theme
      name CDATA   #REQUIRED>
<!ATTLIST superType
      name   CDATA  #REQUIRED>
<!ATTLIST geometry
      type   CDATA  #REQUIRED>
<!ATTLIST roleType
      name   CDATA  #REQUIRED
      relationship IDREF  #REQUIRED>
<!ELEMENT role (featureValue*)>
<!ATTLIST role
      name CDATA #REQUIRED>
<!ELEMENT featureValue EMPTY>
<!ATTLIST featureValue
      idref  CDATA  #REQUIRED>

<!ELEMENT featureUpdate (updateEntity*)>
<!ENTITY % updateEntity "(featureModified|featureDeleted|featureCreated)">
<!ELEMENT featureModified (Feature)>
<!ELEMENT featureDeleted EMPTY >
<!ELEMENT featureCreated (Feature)>

<!ATTLIST featureDeleted
      featureid CDATA #REQUIRED>
```

[Original 2000-07-01, Revised 2000-07-09]