

MERMAID

EU Project No: IST-1999-10637

Supported Data Structures

Deliverable Nos: D2.2
(Rev 2)

June 2001

Report Title:	MERMAID Supported Data Structures
Customer:	European Commission, Directorate-General Information Society, IST Programme
BMT Report no:	13300/D/02.2
Deliverable nos:	D2.2
Report status:	Rev 2
Date:	June 2001
Contact details:	BMT MARINE INFORMATION SYSTEMS LTD Grove House, 7 Ocean Way, Ocean Village, Southampton, Hampshire, SO14 3TJ. United Kingdom. Tel: +44 (0) 2380 232222 Fax: +44 (0) 2380 232891 e-mail: mis@bmtmis.demon.co.uk Website: http://www.bmtmis.com

	Name	Signature	Date
Author:	Paul Taylor		
Approved by:	Dr Andrew Tyler		

This report is commercial-in-confidence. Any information contained herein should not be communicated to any third party without prior permission of BMT Marine Information Systems Ltd.

CONTRIBUTORS

Company	Name
BMT	Paul Taylor
	Paul Goddard
	Chris Rawlings
	Ian Smith
TXT	Matteo Villa
MO	Chris Little
CEDRE	Camille Lecat
IMGW	Wlodek Kryzminski
	Bogusz Biliczweski

DISTRIBUTION LIST

Company	Name	Number of Copies
EU Commission (DG-XIII)	Guy Weets	6
BMT	Paul Taylor	1
TXT	Matteo Villa	1
CEDRE	Vincent Gouriou	1
Met. Office	Jack Hopkins	1
IMGW-OM	Wlodek Kryzminski	1
NC	Koen DeHulsters	1

REVISION

Revision Number	Date	Author	Purpose of Revision
0	26/03/2001	Paul Taylor	Initial Release
1	25/05/2001	Paul Taylor	Inclusion of 2 additional formats (2.2 and 2.3)
2	15/06/01	Ian Smith Paul Taylor	Expansion of section 2, and inclusion of Appendix

TABLE OF CONTENTS

1	INTRODUCTION	6
2	SUPPORTED DATA STRUCTURES	7
2.1	Regular 2D Tidal Harmonic Constituent Data	8
2.2	Irregular 2D Tidal Harmonic Constituent Data	9
2.3	Irregular Residual Current Data	10
2.4	3D Modelled Timeseries Data	11
2.5	Observed Irregular Timeseries Data	12
2.6	Point Location Measured Timeseries.....	14
3	CONCLUSION.....	15
	APPENDIX.....	16

1 Introduction

The data structures that should be supported by the MERMAID system were investigated during the User Requirements phase of the project (see WP2 deliverable; User Requirements Specification - D1). It was decided that there should be no restrictions in the type or format of the data that the system would be able to handle. Providers would therefore be able to register any dataset, regardless of the type or format, with MERMAID, and users (consumers) would be able to search for, and purchase these. However, it was agreed that it would be impossible to develop search and extraction routines that would be capable of interrogating any dataset, regardless of the format. Therefore, a decision was made that a few common data structures would be defined, which MERMAID would specifically support and be capable of interrogating, and retrieving subsets. The structures chosen would be the most suitable for describing the type of data that consumers commonly want to obtain only a small subset of, *i.e.* large datasets covering either large areas, or long time durations (or both), such as the Weather Forecast data produced by the UK Met Office. One of these structures will be the forthcoming WMO format, GRIB2, which is an enhancement on the existing GRIB data format. However, GRIB2 is not due for release until April 2001, and it is envisaged that it will not be in regular use for a few years after this. Therefore search and retrieval algorithms will be developed for the 6 other structures defined below. These are common formats, and as generic as possible. The search and retrieval algorithms for these will be useful in themselves. In addition, they will demonstrate the technology, which will be developed in such a way as to allow easy expansion to support other structures, such as GRIB2, in the future, as and when required.

2 Supported Data Structures

The following six data structures are those that will initially be supported by MERMAID. The appendix provides details of the BINARY structures and coding requirements for each data format.

It should be noted that in all cases, the location specified refers to the centre of the grid cell and that the data value is at the centre of the grid cell.

In addition, in the case of 2D regular gridded formats, the start location (grid reference point) is always the top left corner of the grid (i.e. the north-western-most grid cell). Progression through the grid is from left to right, row by row (i.e. from west to east, north to south) so that the last grid point will be the bottom right corner (south-eastern-most grid cell).

In the case of 3D gridded formats, the start location (grid reference point) is always the top left corner, on the highest vertical layer (i.e. the highest north-western-most grid cell). Progression through the grid is from left to right, row by row, layer by layer (west to east, north to south, vertically downwards). Therefore, the 2D progression is repeated for each horizontal layer. For instance, in a 3D grid that is 10 cells by 10 cells in the horizontal plane, with 20 vertical (depth or height) layers, the total number of grid cells will be 2000. The first data value will be for the top left corner on the highest layer, and the hundredth data value will be for the bottom right corner of the highest layer. The 101st data value will be for the top left corner cell on the next layer down, and so on.

The headers will not contain any character strings, which would make interrogating these fields particularly complex and inefficient. Fields that require textual descriptions, such as parameter names and units, will be enumerated (i.e. defined in a list of options, with each option given a numerical code). These lists will contain limited options initially, but will be extendible for future development.

2.1 Regular 2D Tidal Harmonic Constituent Data

A regular 2D grid of constituent data, independent of time

For instance; 1000 x1500 grid points, on a regular grid with latitude and longitude coordinates, with 12 constituent at each point, and 6 parameters per constituent. The regular grid can therefore be defined by the start point and grid spacing, in a header and inferred (calculated) from this information for each grid point.

Format:

Header

start x grid point (in decimal degrees - i.e. 53.9783°)
start y grid point (in decimal degrees - i.e. -3.5423°)
number of x grid points (i.e. 1000)
number of y grid points (i.e. 1500)
x grid spacing (in decimal degrees - i.e. 0.05°)
y grid spacing (in decimal degrees - i.e. 0.025°)
number of constituents (i.e. 12)
number of parameters (i.e. 6)
amplitude units (enumerated list of options - i.e. m/s, cm/s, kmph, mph or knots)
phase units (enumerated list of options - i.e. degrees or radians)

Data

(Gridded, row by row)

Grid point 1

Constituent 1 Name: ZH,ZG,UH,UG,VH,VG

↓

Constituent 12 Name: ZH,ZG,UH,UG,VH,VG

Grid point 2

Constituent 1 Name: ZH,ZG,UH,UG,VH,VG

↓

Constituent 12 Name: ZH,ZG,UH,UG,VH,VG

Grid Point 3

etc.

2.2 Irregular 2D Tidal Harmonic Constituent Data

An irregular 2D coverage of constituent data, independent of time

For instance; 500 x 700 points, with an irregular spacing, with latitude and longitude co-ordinates, with 6 constituent at each point, and 4 parameters per constituent. Each point must be explicitly defined (in decimal degrees), as the grid is irregular.

Format:

Header

Number of Points: (i.e. 350000)

Region of Influence: (in decimal degrees - i.e. 0.1°)

number of constituents (i.e. 6)

number of parameters (i.e. 4)

amplitude units (enumerated list of options - i.e. m/s, cm/s, kmph, mph or knots)

phase units (enumerated list of options - i.e. degrees or radians)

Data

Location 1 (lat, long – in decimal degrees)

Constituent 1 Name: UH,UG,VH,VG



Constituent 6 Name: UH,UG,VH,VG

Location 2

Constituent 1 Name: UH,UG,VH,VG



Constituent 6 Name: UH,UG,VH,VG

Location 3

etc.

2.3 Irregular Residual Current Data

An irregular coverage of monthly averages of residual current data.

Example:

For instance 453 x 321 points with an irregular spacing of monthly averaged current speed and direction over a period of a year. The region of influence is the minimum distance (in decimal degrees) required to fill in data between points.

Format

Header:

Number of Points: (i.e. 145413)

Region of Influence: (in decimal degrees - i.e. 0.05°)

Magnitude (Speed) unit (enumerated list of options - i.e. m/s, kmph, mph or knots)

Direction (Dir) units (enumerated list of options - i.e. degrees or radians)

Data

Location 1 (lat, long),

Jan Spd, Jan Dir, Feb Spd, Feb Dir, → Nov Spd, Nov Dir, Dec Spd, Dec Dir



Location 145413 (lat, long),

Jan Spd, Jan Dir, Feb Spd, Feb Dir, → Nov Spd, Nov Dir, Dec Spd, Dec Dir

2.4 3D Modelled Timeseries Data

A regular 3D grid of data over a particular period of time (say 1 year). The data measured can be any relevant parameter, and the number of parameters can vary (i.e. Current Speed and Direction, or Air Temp, Air Pressure, Wind Speed and Direction)

Example:

For instance, 400 x 500 x 20 grid points on a regular grid with latitude and longitude co-ordinates. The regular grid can therefore be defined by the start point and grid spacing, in a header and inferred (calculated) from this information for each grid point. The time interval is 1 hour (for instance), with 3 parameters per time step.

Format:

Header

start x grid point (in decimal degrees - i.e. 59.0 degrees)
start y grid point (in decimal degrees - i.e. -5.0 degrees)
start z grid point (i.e. 10m)
number of x grid points (i.e. 400)
number of y grid points (i.e.500)
number of z grid points (i.e. 20)
x grid spacing (in decimal degrees - i.e. 0.05 deg)
y grid spacing (in decimal degrees - i.e. 0.01 deg)
z grid spacing (in metres - i.e. 20m)
start time (seconds from a start reference (i.e. 01.01.1900) - dd.mm.yyyy.hh.mm.ss)
end time (total number of seconds from start time - dd.mm.yyyy.hh.mm.ss)
time interval (in seconds - i.e. 1 hour=3600s)
number of parameters (i.e. 3)
parameter 1 name (P1) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)
parameter 1 unit (enumerated list of options - i.e. m/s, kmph, mph or knots)
↓
parameter n name Pn) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)
parameter n unit (enumerated list of options - i.e. m/s, cm/s, kmph, deg. C, etc.)

Data

(At each time step, the parameters for each grid point is given, before moving on to the next timestep. - hour value runs from 1 at start of timeseries to n at end)

Grid point 1, P1 Value, P2 Value, P3 Value, hour value 1
↓
Grid point n, P1 Value, P2 Value, P3 Value, hour value 1
↓
Grid point 1, P1 Value, P2 Value, P3 Value, hour value n
↓
Grid point n, P1 Value, P2 Value, P3 Value, hour value n

2.5 Observed Irregular Timeseries Data

An irregular coverage of data, with different time periods for each point.

For instance, 500 irregular points, with latitude and longitude co-ordinates. The points cannot be defined implicitly, and the actual co-ordinates are required for each data point. The time interval may change, and each timeseries may start and end at different times. There may be more than one parameter per point and timestep.

Format

Header

No of points (i.e. 500)

Sub-header 1

location 1 (Lat1, Long1)

start time (seconds from a start reference (i.e. 01.01.1900) - dd.mm.yyyy.hh.mm.ss)

end time (total number of seconds from start time - dd.mm.yyyy.hh.mm.ss)

time interval (in seconds - i.e. 3600s)

number of parameters (n)

parameter 1 name (P1) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter 1 unit (enumerated list of options - i.e. m/s, cm/s, kmph, mph or knots)

↓

parameter n name (Pn) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter n unit (enumerated list of options - i.e. m/s, cm/s, kmph, deg. C, etc.)

Data 1

Timestep1, Parameter Values (P1,P2,P3...Pn)

Timestep 2, Parameter Values (P1,P2,P3...Pn)

↓

timestep n, Parameter Values (P1,P2,P3...Pn)

Sub-header 2

location 2 (Lat1, Long2)

start time (seconds from a start reference (i.e. 01.01.1900) - dd.mm.yyyy.hh.mm.ss)

end time (total number of seconds from start time - dd.mm.yyyy.hh.mm.ss)

time interval (in seconds – i.e. 900s)

number of parameters (n)

parameter 1 name (P1) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter 1 unit (enumerated list of options - i.e. m/s, cm/s, kmph, mph or knots)

↓

parameter n name (Pn) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter n unit (enumerated list of options - i.e. m/s, cm/s, kmph, deg. C, etc.)

Data 2

Timestep1, Parameters (P1,P2,P3...Pn)

Timestep 2, Parameters (P1,P2,P3...Pn)



Timestep n, Parameters (P1,P2,P3...Pn)



Sub-header n (500)

Etc

2.6 Point Location Measured Timeseries

A single point location (lat, long) of measured timeseries of one or more parameters.

For example:

Five years of wind history data from the North Sea, at 6 hourly intervals, giving data for 5 parameters

Format

Header

Location (lat, long)

start time (seconds from a start reference (i.e. 01.01.1900) - dd.mm.yyyy.hh.mm.ss)

end time (total number of seconds from start time - dd.mm.yyyy.hh.mm.ss)

Time interval (in seconds - i.e. 21600s)

Number of parameters (i.e. 5)

parameter 1 name (P1) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter 1 unit (enumerated list of options - i.e. m/s, cm/s, kmph, mph or knots)

↓

parameter n name (Pn) (enumerated list of options - i.e. current velocity, wind speed, air temperature, etc.)

parameter n unit (enumerated list of options - i.e. m/s, cm/s, kmph, deg. C, etc.)

Data

Time1, P1 Value, P2 Value, ..., Pn Value

Time2, P1 Value, P2 Value, ..., Pn Value

↓

TimeT, P1 Value, P2 Value, ..., Pn Value

3 Conclusion

The consortium will therefore develop some example search and extraction routines for a limited number of supported formats, familiar to the consortium members, as part of the Data Access Engine component (see WP3 deliverable; Data Access Engine Design – D2.3). These example search and extraction utilities can then be provided free of charge to other data providers for modification, in order to allow the interrogation of other, non-supported data structures.

APPENDIX

This appendix provides details of the code (written in VisualC++) required for each of the six different data structures, including both the headers and the data itself.

The first part (Global Helper Structures) is applicable for all six formats.

It should be noted that in all cases the variable `int` is a 4 byte integer, and `double` is an 8 byte double integer.

```
/*----- Global helper structures -----*/

struct MerTime
{
public:
    int m_day;
    int m_month;
    int m_year;
    int m_hour;
    int m_min;
    int m_sec;
};

struct MerParam
{
    int m_enumParam; // index into param name
    int m_enumUnits; // index into lookup table
};

/*-----
-----*/
```



```
/*-----*/
-----*/
/*--- CMerR2DTHCData: 2.1 Regular 2D Tidal Harmonic Constituent
Data ---*/
/*-----*/
-----*/

class CMerR2DTHCData
{
public:
    CMerR2DTHCData();
    ~CMerR2DTHCData();

//members
public:
//header
    double    m_startX;                // start x grid point
(in degrees)
    double    m_startY;                // start y grid point
(in degrees)
    int       m_nXGridPoints;
    int       m_nYGridPoints;
    double    m_XgridSpacing;          // x grid spacing
(in degrees)
    double    m_YgridSpacing;          // x grid spacing
(in degrees)
    int       m_nConstituents;         // per point
    int       m_nParameters;          // num parameters per
constituent
    int       m_enumAmplitudeUnits;    // index into lookup
table
    int       m_enumPhaseUnits;        // index into lookup
table
// data:- m_nXGridPoints * m_nYGridPoints * m_nConstituents *
m_nParameters * sizeof(double)
    void*    m_pData;
};

/*-----*/
-----*/
```



```
/*-----*/
-----*/
/*-- CMerI2DTHCData: 2.2 Irregular 2D Tidal Harmonic
Constituent Data --*/
/*-----*/
-----*/

class CMerI2DTHCData
{
public:
    CMerI2DTHCData();
    ~CMerI2DTHCData();

//members
public:
//header
    int         m_nPoints;
    double      m_ROI;           // region of
influence (in degrees)
    int         m_nConstituents; // per point
    int         m_nParameters;   // per constituent
    int         m_enumAmplitudeUnits; // index into lookup
table
    int         m_enumPhaseUnits; // index into lookup
table
// data:- m_nPoints * m_nConstituents * m_nParameters *
sizeof(double)
    void*      m_pData;
};

/*-----*/
-----*/
```



```
/*-----*/
-----*/
/*----- CMerI2DRCDData: 2.3 Irregular Residual Current Data
-----*/
/*-----*/
-----*/

struct MerRCDataItem // local helper structure
{
    double m_speed;
    double m_direction;
};

struct MerRCData // local helper structure
{
    MerRCDataItem m_months[12];
};

class CMerI2DRCDData
{
public:
    CMerI2DRCDData();
    ~CMerI2DRCDData();

//members
public:
//header
    int m_nPoints;
    double m_ROI; // region of
influence (in degrees)
    int m_enumMagnitudeUnits; // index into
lookup table
    int m_enumDirectionUnits; // index into
lookup table
// data:- m_nPoints * sizeof(MerRCData)
    void* m_pData;
};

/*-----*/
-----*/
```

```
/*-----*/
-----*/
/*----- CMerR3DTSDData: 2.4 3D Modelled Timeseries Data -----*/
-----*/
/*-----*/
-----*/

class CMerR3DTSDData
{
public:
    CMerR3DTSDData();
    ~CMerR3DTSDData();
//members
public:
//header
    double    m_startX;                // start x grid point
(in degrees)
    double    m_startY;                // start y grid point
(in degrees)
    double    m_startZ;                // start z grid point
(in metres)
    int       m_nXGridPoints;
    int       m_nYGridPoints;
    int       m_nZGridPoints;
    double    m_XgridSpacing;          // x grid spacing
(in degrees)
    double    m_YgridSpacing;          // y grid spacing
(in degrees)
    double    m_ZgridSpacing;          // z grid spacing
(in metres)
    MerTime   m_startTime;
    MerTime   m_endTime;
    int       m_timeInterval;          // (in seconds)
    int       m_nParameters;          // num parameters per
constituent
// parameters
    void*     m_pParameters;           // m_nParameters *
sizeof(MerParam)
// data:- nTimesteps * (m_nXGridPoints * m_nYGridPoints *
m_nZGridPoints * m_nParameters * sizeof(double))
    void*     m_pData;
};

/*-----*/
-----*/
```



```
/*-----*  
-----*/  
/*----- CMerOITSData: 2.5 Observed Irregular Timeseries Data  
-----*/  
/*-----*  
-----*/  
  
struct MerOITSSubData // local helper structure  
{  
    double    m_locX;                // location x grid  
point (in degrees)  
    double    m_locY;                // location y grid  
point (in degrees)  
    MerTime   m_startTime;  
    MerTime   m_endTime;  
    int       m_timeInterval;        // (in seconds)  
    int       m_nParameters;        // num parameters per  
location  
// parameters  
    void*     m_pParameters;        // m_nParameters *  
sizeof(MerParam)  
// data:- nTimesteps * (m_nParameters * sizeof(double))  
    void*     m_pData;  
};  
  
class CMerOITSData  
{  
public:  
    CMerOITSData();  
    ~CMerOITSData();  
//members  
public:  
//header  
    int       m_nPoints;  
// data:- m_nPoints occurrences of MerOITSSubData  
    void*     m_pData;  
};  
/*-----*  
-----*/
```



```
/*-----*/
-----*/
/*----- CMerPLMTSData: 2.6 Point Location Measured
Timeseries -----*/
/*-----*/
-----*/

class CMerPLMTSData
{
public:
    CMerPLMTSData();
    ~CMerPLMTSData();

//members
public:
//header
    double    m_locX;                // location x grid
point (in degrees)
    double    m_locY;                // location y grid
point (in degrees)
    MerTime   m_startTime;
    MerTime   m_endTime;
    int       m_timeInterval;        // (in seconds)
    int       m_nParameters;        // num of parameters
per location
// parameters
    void*     m_pParameters;        // m_nParameters *
sizeof(MerParam)
// data:- nTimesteps * (m_nParameters * sizeof(double))
    void*     m_pData;
};

/*-----*/
-----*/
```