FFI-rapport 2009/00188

LYBIN 5.0 – interface description

Elin Dombestein

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

1 February 2009

FFI-rapport 2009/00188

356901

P: ISBN 978-82-464-1549-9
E: ISBN 978-82-464-1550-5

Keywords

Akustisk deteksjon

Modellering og simulering

Programmering (Databehandling)

Approved by

Elin Dombestein	Project Manager
Elling Tveit	Director of Research
John-Mikal Størdal	Director

English summary

The acoustic ray trace model LYBIN is a well established and frequently used sonar prediction model owned by the Norwegian Defence Logistic Organisation. The model is used onboard navy vessels as well as in training situations on shore. LYBIN has become an important tool in both planning and evaluation of maritime operations, and earlier versions are already integrated in combat system software, tactical decision aids and tactical trainers.

The purpose of this report is to describe the software interfaces needed for the integration of the current version of LYBIN in other software applications. Different from earlier, third parties can now integrate LYBIN in their software without needing access to the source code.

LYBIN is now implemented as a softwaremodule (COM) for the Microsoft Windows platform. In addition there already exists a graphical user interface which can be used together with the COM module to build a stand alone executive application. An implementation as a COM module makes LYBIN suitable for integration with other applications, and enables LYBIN to interact with other applications such as other mathematical models, web services, geographical information systems and more.

The COM module has two different interfaces for data exchange with other software. The two interfaces are the binary interface and the eXtensible Markup Language (XML) interface. The binary interface enables fast transportation of large amounts of data to and from LYBIN. The XML interface is not as fast, but is more robust because the format of the input files is not so strict. The XML interface discards any parts of the input file it does not recognize.

All parameters and data sets that can be passed to and from LYBIN are described in this report. Examples of programming code for integration of LYBIN are also included. What goes on inside the interfaces and how the acoustic modeling is performed will not be discussed.

Sammendrag

Den akustiske strålegangsmodellen LYBIN er en etablert og mye brukt sonar ytelsesmodell som eies av Forsvarets Logistikkorganisasjon. Modellen brukes både ombord på marinefartøy og i treningssituasjoner på land. LYBIN er blitt et viktig verktøy både i planlegging og evaluering av maritime operasjoner, og tidligere versjoner er allerede integrert i programvare for kampsystemer, taktiske beslutningsstøtte og taktiske trenere.

Hensikten med denne rapporten er å beskrive de programmeringsgrensesnittene som trengs for å kunne integrere dagens versjon av LYBIN i andre programvareapplikasjoner. I motsetning til tidligere er det nå mulig for andre å integrere LYBIN i deres programvare uten å ha tilgang til LYBINs kildekode.

LYBIN er nå implementert som en software modul (COM modul) for Microsoft Windows plattformen. I tillegg eksisterer det allerede et grafisk brukergrensesnitt som sammen med COM modulen kan brukes for å bygge en frittstående eksekverbar applikasjon. Å lage LYBIN som en COM modul gjør LYBIN egnet for integrasjon med andre applikasjoner, og muliggjør at LYBIN kan samhandle med andre applikasjoner som matematiske modeller, web-tjenester, geografiske informasjonssystemer med mer.

COM modulen har to ulike grensesnitt for datautveksling med annen programvare. De to grensesnittene er det binære grensesnittet og eXtensible Markup Language (XML) grensesnittet. Det binære grensesnittet muliggjør rask transport av store mengder data til og fra LYBIN. XML grensesnittet er ikke like raskt, men er mer robust fordi formatet til inputfilene ikke er så rigid. XML grensesnittet forkaster de delene av inputfila det ikke gjenkjenner.

Alle parametere og datasett som kan sendes til og fra LYBIN er beskrevet i denne rapporten. Noen eksempler på programkode for integrasjon av LYBIN er også inkludert. Det som skjer innenfor grensesnittene og hvordan den akustiske modelleringen er gjort vil ikke bli omtalt.

Contents

1 Introduction

The acoustic ray trace model LYBIN is a well established and frequently used sonar prediction model owned by the Norwegian Defence Logistic Organisation (NDLO). The model is used onboard navy vessels as well as in training situations on shore. LYBIN has become an important tool in both planning and evaluation of maritime operations.

LYBIN is a range dependent incoherent two-dimensional model. Several thousand rays are simulated traversing the water volume. Upon hitting the sea surface and sea bed, the rays are diffracted and exposed to loss mechanisms. Even in the water volume itself, rays are affected and energy is lost. LYBIN estimates the probability of detection for a given target based on the calculated transmission loss, reverberation and target strength. LYBIN is a robust, user friendly and fast acoustic ray-trace simulator. The physics behind the model are described in more detail in [1].

On behalf of NDLO, the Norwegian Defense Research Institute (FFI) has been responsible for testing, evaluation and further development of LYBIN since the year 2000. During this period, two new versions of LYBIN have been released: LYBIN 3.0 and LYBIN 4.0. LYBIN 5.0 will be released in august 2009. This document describes the interface of LYBIN 5.0.

LYBIN 5.0 is implemented as a COM module for the windows platform. In addition there exists a graphical user interface which can be used together with the COM module in order to build a stand-alone executive application. The COM module enables LYBIN to interact with other applications, such as other mathematical models, web services, geographical information systems and more.

The COM module has two different interfaces for data exchange. The two interfaces are the *binary interface* and the *eXtensible Markup Language* (XML) interface. Advantages and disadvantages of both will be discussed in the report.

The report is divided in three major parts. The first part describes the overall configuration of LYBIN. In the second part, the binary interface is described in detail, while the third part treats the XML interface. Only the part of the XML interface that is new to LYBIN 5.0 is described in this report.

2 The configuration of LYBIN 5.0

LYBIN 5.0 is divided into two separate parts: the calculation kernel and the graphical user interface. This separation enables LYBIN to interact with other applications, such as mathematical models, web services, geographic information systems, and more.

The graphical user interface represents the classical LYBIN application, where LYBIN is used as stand-alone software. Environmental data and information about the sonar and sonar platform are entered to the calculation kernel, by the operator through the graphical user interface. The calculation results are thereafter plotted by the graphical user interface and can either be copied as a picture or saved as a file of binary data for later use by external programs. A picture of the traditional LYBIN graphical user interface is shown in Figure 2.1.

The stand-alone calculation kernel enhances the potential applicability of LYBIN by enabling connectivity and communication between systems. The LYBIN kernel can be integrated with external applications, and both input and calculation results can be handled automatically from outside applications.

One example of integration of the LYBIN kernel is calculation of sonar coverage. The environmental data are fed to the LYBIN kernel and calculations started from the external application, while the calculation results are displayed in the external program. The output from such an application is shown in Figure 2.2, where calculated signal excess is plotted in the geographical information system Maria [2]. The integration between LYBIN and Maria is described in more detail in [3] and [4].

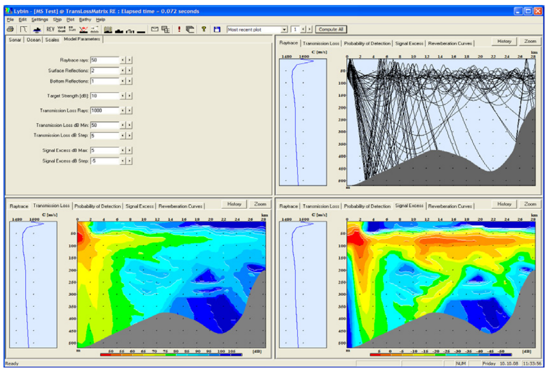

Figure 2.1 LYBINs traditional standard graphical user interface.

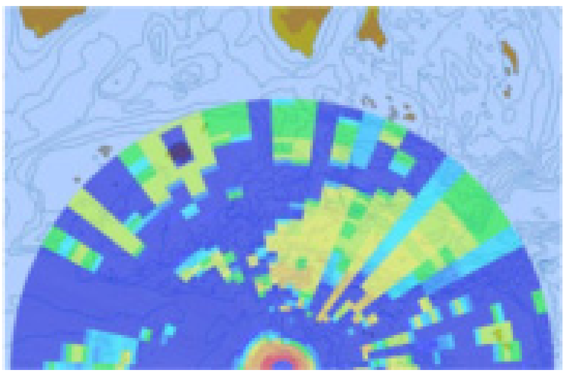

Figure 2.2 LYBIN integrated with the geographical information system Maria.

2.1 The COM object

The calculation kernel is implemented as a COM module for the windows platform. The COM object has a series of interfaces with their properties and functions available. The interfaces are listed in Table 2.1.

By using a .NET language such as C#.NET, C++.NET, Visual Basic .NET, Fortran.NET, Pyton.NET etc. all the interfaces are available through the same class object. All the code examples in this report are written in C#.NET using the object LybinModelComBinClass. For the XML interface there is a corresponding class object called LybinModelComClass .

Interface name	Description
IEnvironment	Environmental data.
ILybinModelCom	Start calculation, get results and see the calculation parameters in XML form.
ILybinModelComBin	Start calculation, get results and see the calculation parameters in binary form.
IModelData	Parameters controlling the calculations such as accuracy, which datasets to be used etc.
IOcean	Parameters describing the media (ocean) and the assumed target.
IPlatform	Description of the platform holding the sensor.
IPulse	Parameters describing the sensor pulse.
ISensor	Parameters describing the sensor.

Table 2.1 The LYBIN COM object interfaces.

The LYBIN COM object has two different interfaces for data exchange with other applications, where both have their individual advantages.

The XML interface uses extensible markup language (XML) to control and manage the information going to and from LYBIN. XML is an open standard with a simple syntax and an unambiguous structure. The XML interface is very robust since it discards any parts of the code it does not recognize. The data exchange will not halt due to format errors in the input file. Additional information can also be included in the XML files without interrupting the data exchange between the LYBIN kernel and the external application.

The binary interface provides a faster data exchange than the XML interface. The binary file format is more rigorous though. Deviations from the defined format will lead to failures in the data transfer process. The binary interface contains both method calls and variable calls to access the kernel.

2.2 The calculation kernel

The calculation kernel is the core of LYBIN. In the kernel, the environmental and sonar information are processed and all the mathematical calculations are performed. If we take a look inside the kernel, we will see that it is divided into three separate parts. We have the data layer, the calculation layer and the result layer. Each of these is described in more details below. A schematic description of the LYBIN calculation kernel is shown in Figure 2.3.

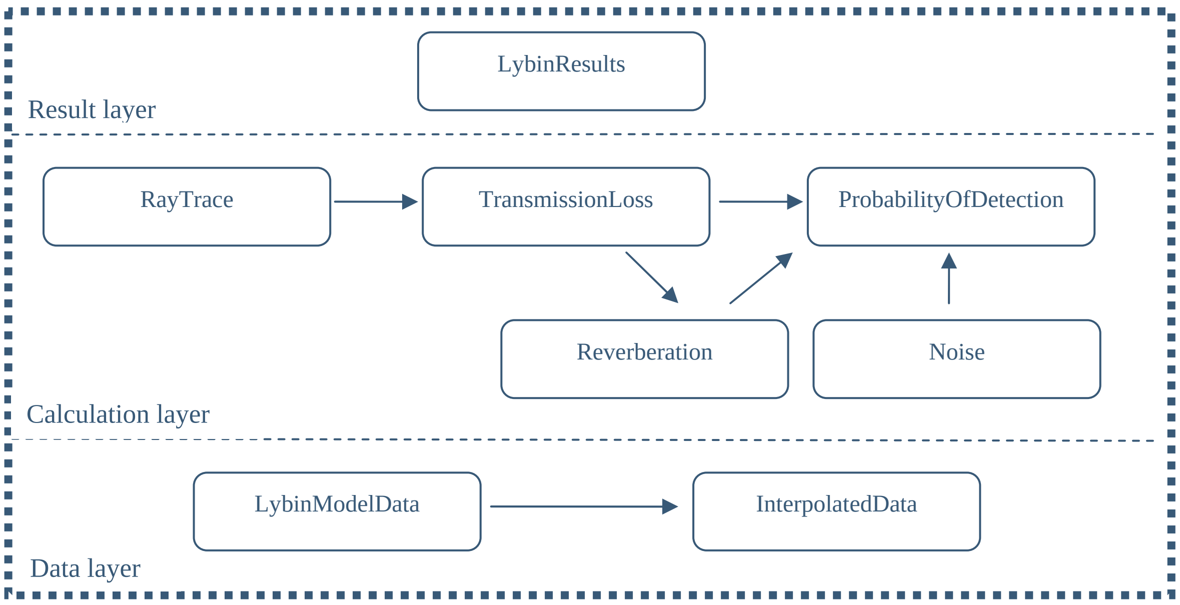

Figure 2.3 Schematic description of the LYBIN calculation kernel.

2.2.1 The data layer

The data layer stores all the data given to the model. All the datasets forming the basis of the acoustic calculations in LYBIN are organized in classes. A class diagram for the LYBIN input data model is showed in Figure 2.4 and each of these classes are described shortly in Table 2.2.

LybinModelData has two members: the environment class and the platform class. The environment class is an assembly of all the environmental data LYBIN uses in the calculations. The platform class holds all the information about the sonar and the sonar platform.

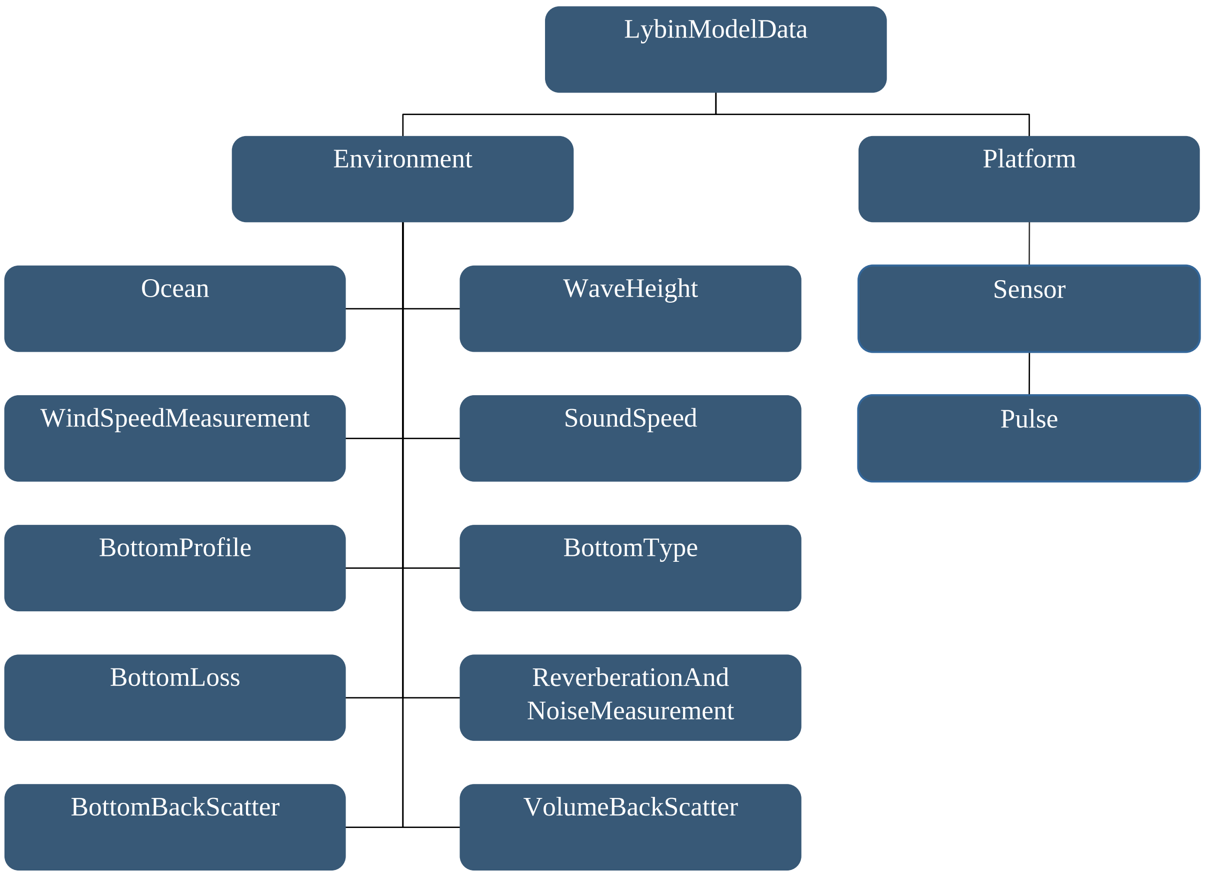

Figure 2.4 The LYBIN input data model (Class diagram).

Class name	Description
BottomBackScatter	Range dependent bottom back scatter values as a function of each ray’s grazing angle with the bottom.
BottomLoss	Range dependent bottom loss values as a function of each ray’s grazing angle with the bottom.
BottomProfile	Single measurements of depth as a function of range from the sensor.
BottomType	Range dependent bottom types ranging from 0-10. The bottom type is transformed into bottom loss before it is used in model calculations.
Environment	An assembly class for all environmental data: holds no parameters of its own.
Model	Parameters controlling the calculations such as accuracy, which datasets to be used etc.
Ocean	Parameters describing the media (ocean) and the assumed target, such as ambient noise, pH, surface scatter, target strength and ship density.
Platform	Description of the platform (ship or helicopter) holding the sensor. Contains platform speed and noise.
Pulse	Parameters describing the sensor pulse: its form, length, bandwidth etc.
ReverberationAndNoiseMeasurement	Range dependent total reverberation and noise data.
Sensor	Parameters describing the sensor (sonar) such as depth, tilt, frequency etc.
SoundSpeed	Range dependent sound speed, temperature and salinity measurements as function of depth.
VolumeBackScatter	Range dependent volume back scatter values.
WaveHeight	Range dependent wave height measurements.
WindSpeedMeasurement	Range dependent wind speed measurements.

Table 2.2 LYBIN classes containing input data.

2.2.1.1 Range dependent data

LYBIN is able to handle range dependent environments. In LYBIN, range dependent environmental data can be given for a set of specific locations (geographical points), or be specified for certain range intervals from the transmitting sonar.

When the environmental properties are entered for a discrete set of locations (ranges), LYBIN will create intervening values as a function of range using interpolation. If no environmental descriptions is given at zero range, LYBIN will substitute the data for the nearest range available. Likewise if data at maximum range are missing.

Except for BottomProfile and ReverberationAndNoiseMeasurement, the range dependent data are given with start and stop values to indicate their range of validity. In this context, we call these datasets, with start and stop related to a value (or sets of values), for an object. A range dependent object can contain one or more values with their range of validity. The structure of a range dependent object with start and stop range is shown in Figure 2.5. The possible numbers of values to be used in the calculation are only limited by the calculation accuracy.

The start and stop functionality provides great flexibility in defining the environmental range dependent properties. By setting start and stop to the same range, the values will be considered to belong to a point in space, and LYBIN will use interpolation to produce data for intermediate ranges points. The start and stop functionality might be utilized to illustrate meteorological or oceanographic fronts, entering ranges with finite ranges of validity to each side of the front, and separating the sets by any small distance, across which the conditions will change as abruptly as the user intends. In between these two extreme choices all combination of these are possible to use.

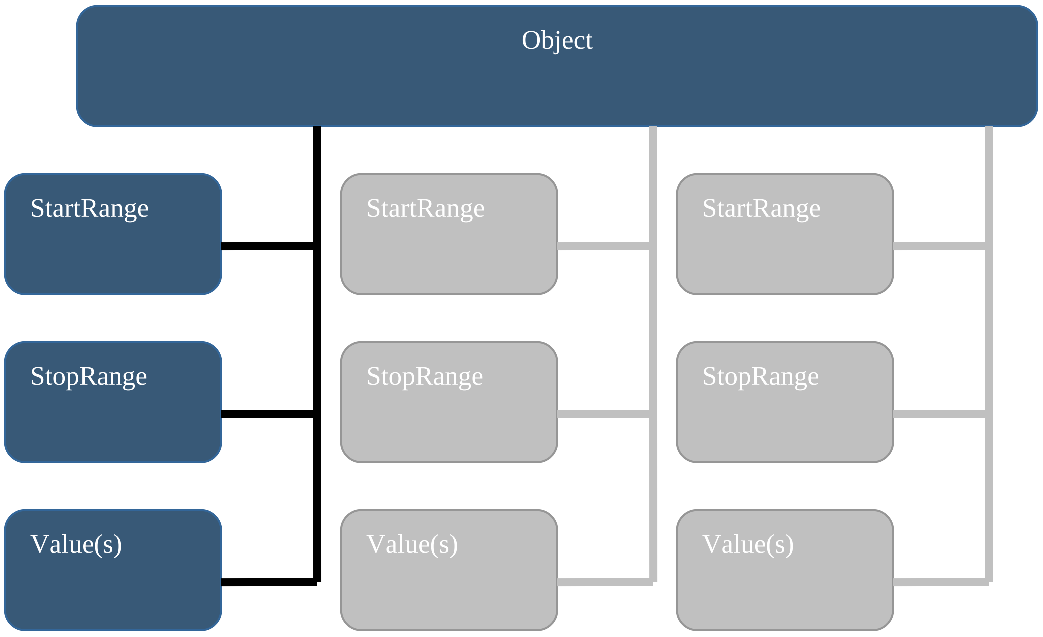

Figure 2.5   Schematic description of a range dependent object with start and stop parameters.

2.2.2 The calculation layer

In the calculation layer, all the acoustic calculations are performed. First the ray trace is calculated. The intensity of all the rays is then summed up within every calculation cell in order to compute the transmission loss. The reverberation is found based on the backscattering properties and the transmission loss at the sea surface, bottom and volume. Noise is calculated as the sum of the ambient noise in the sea and the sonar self noise. Finally the probability of detection is calculated based on target echo strength, transmission loss, reverberation and noise.

2.2.3 The result layer

The calculation results are stored in the result layer. All the data sets in the result layer are listed below:

- Ray path
- Transmission time
- Transmission loss from sonar to target
- Transmission loss from target to sonar
- Impulse response
- Noise
- Surface reverberation
- Volume reverberation
- Bottom reverberation
- Total reverberation
- Signal excess
- Probability of detection

3 Description of the binary interface

This section describes the binary interface in detail. First the input data are described, then how to initiate a calculation and finally how to access the results.

The binary interface has its advantage in the fast data exchange. This becomes important in the cases where large amounts of data are transferred to or from the LYBIN kernel, examples being detailed bathymetry data or detailed range dependent oceanographic data.

3.1 Input data

Every class below LybinModelData shown in Figure 2.3 is discussed in this section. The access methods and variables are described. Some code examples are also included.

3.1.1 LybinModelData class

The LybinModel class contains parameters controlling the acoustic calculations: the resolution of the calculation, what type of calculation to be performed and so on. All the parameters in LybinModelData class are listed in Table 3.1, and the access functions connected to the LybinModelDataClass are described in Table 3.2.

Parameter	Type	Default value	Unit
BottomReverberationCalculation *Tells LYBIN whether to calculate the bottom reverberation or not.* *False: Do not calculate bottom reverberation.* *True: Do calculate bottom reverberation.*	Boolean	true	
DepthCells *Number of depth cells in the calculation output.*	Integer	50	
DepthCellSize *Size of the depth cells in the calculation output.*	Double	6	Meters
DepthScale *Maximum depth in the calculation.*	Double	300	Meters
DepthSteps *Number of depth steps to be used during the calculation.*	Integer	1000	
DepthStepSize *Size of the depth steps to be used during the calculation.*	Double	0.3	Meters
ImpulseResponseCalculation *Tells LYBIN whether to calculate impulse response or not.* *False: Do not calculate impulse response.* *True: Calculate impulse response.*	Boolean	false	
ImpulseResponseRange *The range of point that the impulse response will be calculated from.*	Double	0	Meters
ImpulseResponseDepth *The depth of point that the impulse response will be calculated from.*	Double	0	Meters
MaxBorderHits *Maximum number of boundaryhits allowed before a ray is terminated. A boundary is either the sea surface of the sea bottom.*	Integer	5000	
ModelData *Total data model holding all the input parameters to be used in the calculation. The model data are returned as an XML string.*	String		
NoiseCalculation *Tells LYBIN whether to calculate the noise or not.* *False: Do not calculate noise.* *True: Do calculate noise.*	Boolean	true	
PassiveCalculation *Tells LYBIN whether to perform calculations for active or passive sonars.*	Boolean	false	

Parameter	Type	Default value	Unit
ProbabilityOfDetectionCalculation *Tells LYBIN whether to calculate the probability of detection or not.* *False: Do not calculate the probability of detection.* *True: Do calculate the probability of detection.*	Boolean	true	
RangeCells *Number of range cells in the calculation output.*	Integer	50	
RangeCellSize *Size of the range cells in the calculation output.*	Double	200	Meters
RangeScale *Maximum range in the calculation*.	Double	10000	Meters
RangeSteps *Number of range steps to be used during the calculation.*	Integer	500	
RangeStepSize *Size of the range steps to be used during the calculation.*	Double	20	Meters
RayTraceCalculation *Tells LYBIN whether to calculate the total ray trace or not.* *False: Do not calculate the total ray trace.* *True: Calculate the total ray trace.*	Boolean	true	
SignalExcessCalculation *Tells LYBIN whether to calculate the signal excess or not.* *False: Do not calculate the signal excess.* *True: Do calculate the signal excess.*	Boolean	true	
SignalExcessConstant *Parameter affecting the relation between signal excess and probability of detection.*	Double	3	
SurfaceReverberationCalculation *Tells LYBIN whether to calculate the surface reverberation or not.* *False: Do not calculate surface reverberation.* *True: Do calculate surface reverberation.*	Boolean	true	
TerminationIntensity *Each ray is terminated when its intensity falls below this value.*	Double	1e-16	
TransmissionLossFromTargetCalculation *Tells LYBIN whether to calculate the transmission loss from target to sonar or not.* *False: Do not calculate the transmission loss from target to sonar.* *True: Do calculate the transmission loss from target to sonar.*	Boolean	true	

Parameter	Type	Default value	Unit
TransmissionLossToTargetCalculation *Tells LYBIN whether to calculate the transmission loss from sonar to target or not.* *False: Do not calculate the transmission loss from sonar to target.* *True: Do calculate the transmission loss from sonar to target.*	Boolean	true	
TravelTimeAngleRes *The distance in degrees between the start angles of the rays to be used in the travel time calculation.*	Double	1	Degrees
TravelTimeCalculation *Tells LYBIN whether to calculate travel time or not.* *False: Do not calculate travel time.* *True: Calculate travel time.*	Boolean	false	
TRLRays *Number of rays to be used in the transmission loss calculation.*	Integer	1000	
TypeOfRevNoiseCalculation *Enumerator used to control how the calculation of reverberation is performed:* *0: Calculate bottom reverberation from bottom types* *1: Calculate bottom reverberation from back scatter values* *2: Use measured reverberation and noise data*	Enumeration	0	
UseMeasuredBottomLoss *Tells the model how to calculate bottom loss:* *False: Use bottom types to calculate bottom loss* *True: Use measured or supplied bottom loss values*	Boolean	false	
VisualRayTraceCalculation *Tells LYBIN whether to calculate a ray trace plot for visualisation or not.* *False: Do not calculate ray trace for visualisation.* *True: Calculate ray trace for visualisation.*	Boolean	false	
VisualSurfaceHits *Number of surface hits alloved in the ray trace plot.*	Integer	2	
VisualBottomHits *Number of bottom hits alloved in the ray trace plot.*	Integer	1	
VisualNumRays *Number of rays in the visual ray plot.*	Integer	50	

Parameter	Type	Default value	Unit
VolumeReverberationCalculation *Tells LYBIN whether to calculate the volume reverberation or not.* *False: Do not calculate volume reverberation.* *True: Do calculate volume reverberation.*	Boolean	true	

Table 3.1 Parameters in the LybinModelData class.

Function	Type	Unit of input parameters
ChangeModelData(string xmlData) *Send in the complete XML LYBIN dataset as one string.*	Void	
GetCurrentModelData(out string modelData) *Get the complete XML LYBIN dataset as one string.*	Void	
SetDepthCellSizeAndDepthSteps(double cellSize, int steps) *Set the depth cell size and the number of depth steps. This setting will overrule all earlier depth settings affecting the calculation precision.*	Void	**cellSize**: meters
SetDepthScaleAndDepthCells(double scale, int cells) *Set the depth scale and the number of depth cells. This setting will overrule all earlier depth settings affecting the calculation precision.*	Void	**scale**: meters
SetDepthScaleAndDepthCellSize(double scale, double cellSize) *Set the depth scale and the depth cell size. This setting will overrule all earlier depth settings affecting the calculation precision.*	Void	**scale**: meters, **cellSize**: meters
SetDepthScaleAndDepthSteps(double scale, int steps) *Set the depth scale and the number of depth steps. This setting will overrule all earlier depth settings affecting the calculation precision.*	Void	**scale**: meters
SetImpulseResponsePosition(double range, double depth) *Set the range and depth of point that the impulse response will be calculated from.*	Void	**Range**: meters, **Depth**: meters
SetRangeCellSizeAndRangeSteps(double cellSize, int steps) *Set the range cell size and the number of range steps. This setting will overrule all earlier range settings affecting the calculation precision.*	Void	**cellSize**: meters
SetRangeScaleAndRangeCells(double scale, int cells) *Set the range scale and the number of range cells. This setting will overrule all earlier range settings affecting the calculation precision.*	Void	**scale**: meters
SetRangeScaleAndRangeCellSize(double scale, double cellSize) *Set the range scale and the range cell size. This setting will overrule all earlier range settings affecting the calculation precision.*	Void	**scale**: meters, **cellSize**: meters

Function	Type	Unit of input parameters
SetRangeScaleAndRangeSteps(double scale, int steps) *Set the range scale and the number of range steps. This setting will overrule all earlier range settings affecting the calculation precision.*	Void	**scale**: meters

Table 3.2 Functions in the LybinModelData class.

3.1.1.1 Calculation switches

In order to optimize the calculation time, LYBIN has calculation switches. They are all listed in Table 3.2. The switches control what part of the total calculation to be performed. Fot instance, if one is only interested in the transmission loss, there is no need to calculate the total signal excess.

There are dependencies inbetween the calculation switches. If one switch is turned on, all the calculation swithses for the calculations nesessary to perform this calculation will automatically be turned on. The hierarcical dependency inbetween the calculation switches are shown in Figure 3.1

3.1.1.2 Switches with dependencies to specific datasets

Both TypeOfRevNoiseCalculation and UseMeasuredBottomLoss can make LYBIN use sertain datasets in stead of predefined default values. In order to follow these demands, the spesified datasets must be sent into LYBIN. If LYBIN can not find these datasets, the switsces will be set back to default values. For both TypeOfRevNoiseCalculation and UseMeasuredBottomLoss, default values means using the predefined bottom types to calculate respectively bottom reverberation and bottom loss.

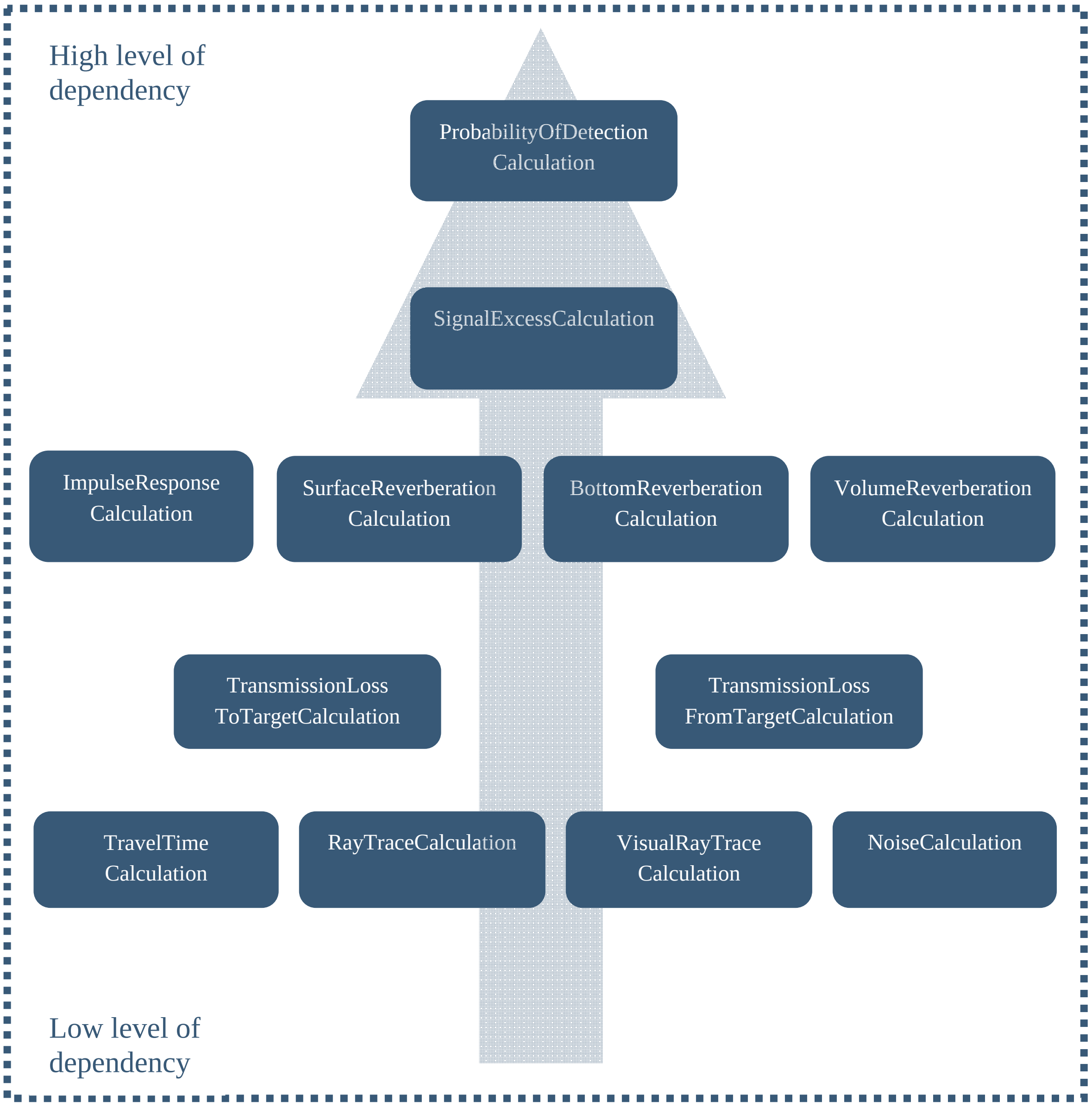
High level of dependency
ProbabilityOfDetection Calculation
SignalExcessCalculation
ImpulseResponse Calculation
SurfaceReverberation Calculation
BottomReverberation Calculation
VolumeReverberation Calculation
TransmissionLoss ToTargetCalculation
TransmissionLoss FromTargetCalculation
TravelTime Calculation
RayTraceCalculation
VisualRayTrace Calculation
NoiseCalculation
Low level of dependency

Figure 3.1 The hierarchical dependency between the calculation switches. The higher on the arrow, the more dependencies.

3.1.2 Environment class

The environment class does not hold any functions or parameters of its own. It is only an assembly class for all the classes holding environmental data.

3.1.3 Ocean class

The parameters in the ocean class represent the ocean environment and targets within the sea. All the parameters in the ocean class are listed in Table 3.3. There is no access functions connected to the ocean class.

Parameter	Type	Default value	Unit
AmbientNoiseLevel *Noise from ambient sources.*	Double	50	dB
AmbientNoiseLevelPassive *Noise from ambient sources to be used in calculations for passive sonar.*	Double	50	dB
PH *pH level in the sea water.*	Double	8	
ShipDensity *Density of ship traffic in the area of the calculation. The ship density can vary from 1 (low) to 7 (high).*	Double	4	
SurfaceScatterFlag *True: Surface reflected ray angles will be modified in order to simulate rough sea scattering.* *False: Rays hitting the sea surface will be reflected specularly, as from a perfectly smooth surface.*	Boolean	true	
TargetStrength *Target echo strenght.*	Double	10	dB
TargetSpeed *Target speed.*	Double	10	Knots

Table 3.3 Parameters in the Ocean class.

3.1.4 WindSpeedMeasurement class

The WindspeedMeasurement class only has one accessible parameter, the WindSpeedMeasurement, which is listed in Table 3.4.

Parameter	Type	Default values	Units
WindSpeedMeasurment *Wind speed in the area of calculation.*	Object *(Double[x,3])*	(0, 0, 0) *(start, stop, value)*	(Kilometres, Kilometres, Meters/Second)

Table 3.4 Parameters in the WindSpeedMeasurement clas:, x can be any number.

An example of how WindSpeedMeasurement can be used is shown in the C# code example below. In the example, the measured wind speed is 2 meters/second from 0 to 5 kilometers, and 4 meters/second from 5 to 10 kilometers.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Wind
double[,] ws = new double[2, 3];
ws[0, 0] = 0;
ws[0, 1] = 5;
ws[0, 2] = 2;
ws[1, 0] = 5;
ws[1, 1] = 10;
ws[1, 2] = 4;

Lybin.WindSpeedMeasurment = ws;
```

3.1.5 WaveHeight class

The WaveHeight class only has one accessible parameter, the WaveHeight, which is listed in Table 3.5.

Parameter	Type	Default value	Unit
WaveHeight *Wave height in the area of calculation.*	Object (Double[x,3])	(0, 0, 0) *(start, stop, value)*	(Kilometres, Kilometres, Meters)

Table 3.5 Parameters in the WaveHeight class, x can be any number.

An example of how WaveHeight can be used is shown below. In the example the wave height is 1 meter from 0 to 5 kilometers, and 2 meters from 5 to 10 kilometres.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Wave height
double[,] wh = new double[2, 3];
wh[0, 0] = 0;
wh[0, 1] = 5;
wh[0, 2] = 1;
wh[1, 0] = 5;
wh[1, 1] = 10;
wh[1, 2] = 2;

Lybin.WaveHeight = wh;
```

3.1.6 SoundSpeed class

The SoundSpeed class handles the sound speed in the water volume. The sound speed is a function of both range and depth. Since the sound speed is most often measured as depth profiles, the sound speed profile is the basis of the SoundSpeed class. The profile is localized in range using the start and stop parameters.

The profile can contain both temperature, salinity and sound speed for a given set of depths. If all three are given, the sound speed entries will be ignored and LYBIN will calculate the sound speeds from the tabulated depths, temperatures and salinities. If two of the three are given, LYBIN will estimate the remaining one based on the other two. If only sound speeds *or* temperatures are available, LYBIN will estimate the missing profile (temperature or sound speed) using a user defined salinity profile. Sound speed for intermediate depths are computed using interpolation.

There is only one parameter in the SoundSpeed class, the SoundSpeedProfileCount, given in Table 3.6. The functions in the SoundSpeed class are given in Table 3.7.

Parameter	Type	Default value	Unit
SoundSpeedProfileCount *Number of sound speed profiles.*	Integer	1	

Table 3.6 Parameters in the SoundSpeed class.

Function	Type	Unit of input parameters
AddSalinityProfile(int start, int stop, object profile) *Add another salinity profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	**start**: kilometres **stop**: kilometres **<u>profile:</u>** **depth**: meters **salinity**: parts per thousand **sound speed**: meters/second **temperature**: degrees Celsius
AddSoundSpeedProfile(int start, int stop, object profile) *Add another sound speed profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
AddSoundSpeedAndSalinityProfile(int start, int stop, object profile) *Add another sound speed and salinity profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
AddSoundSpeedAndTempProfile(int start, int stop, object profile) *Add another sound speed and temperature profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
AddSoundSpeedTempAndSalinityProfile(int start, int stop, object profile) *Add another sound speed, temperature and salinity profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
AddTempAndSalinityProfile(int start, int stop, object profile) *Add another temperature and salinity profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
AddTempProfile(int start, int stop, object profile) *Add another temperature profile. This function can only be used after the first profile has been added with one of the SetFirstProfile functions.*	Void	
GetSoundSpeedProfile(int index, out int start, out int stop, out object profile) *Get the sound speed profile corresponding to the given index.*	Void	
SetFirstSalinityProfile(int start, int stop, object profile) *Set the first salinity profile.*	Void	
SetFirstSoundSpeedProfile(int start, int stop, object profile) *Set the first sound speed profile.*	Void	
SetFirstSoundSpeedAndSalinityProfile(int start, int stop, object profile) *Set the first sound speed and salinity profile.*	Void	

Function	Type	Unit of input parameters
SetFirstSoundSpeedAndTempProfile(int start, int stop, object profile) *Set the first sound speed and temperature profile.*	Void	
SetFirstSoundSpeedTempAndSalinityProfile (int start, int stop, object profile) *Set the first sound speed, temperature and salinity profile.*	Void	
SetFirstTempAndSalinityProfile(int start, int stop, object profile) *Set the first temperature and salinity profile.*	Void	
SetFirstTempProfile(int start, int stop, object profile) *Set the first temperature profile.*	Void	

Table 3.7 Functions in the SoundSpeedClass.

An example of how some of the sound speed functions can be used is shown below. In the example, the first sound speed profile is set. At the range from 0 to 20 kilometres, LYBIN is to use the profile given by the sound speed 1480 m/s at 5 meters depth and the sound speed 1500 m/s at 100 meters depth. At the end of the example, the first sound speed profile is retrieved from LYBIN. This profile contains calculated temperature, salinity and sound speed as used in the calculations.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Set the first sound speed profile
double[,] ssp = new double[2, 2];
ssp[0, 0] = 5;
ssp[0, 1] = 1480;
ssp[1, 0] = 100;
ssp[1, 1] = 1500;
Lybin.SetFirstSoundSpeedProfile(0, 20, ssp);

// Get the first SoundSpeedProfile
int index = 0;
int start, stop;
object profile = new object();
Lybin.GetSoundSpeedProfile(index, out start, out stop, out profile);
```

3.1.7 BottomProfile class

The BottomProfile class only has one accessible parameter, the BottomProfile, which is listed in Table 3.8. The BottomProfile can consist of any number points in range with their corresponding bottom depths.

Parameter	Type	Default value	Unit
BottomProfile *Depth of bottom as function of range.*	Object (Double[x,2])	(0, 280) (*range, depth*)	(Metres, Meters)

Table 3.8 Parameter in the bottom profile class, x can be any number.

An example on how the BottomProfile can be used is shown below. In the example, two points are inserted. The first is the depth 300 meters at a range of 0 meter. The second is the depth 380 meters at a range of 1000 meters.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Bottom
double[,] bp = new double[2, 2];
bp[0, 0] = 0;
bp[0, 1] = 300;
bp[1, 0] = 1000;
bp[1, 1] = 380;

Lybin.BottomProfile = bp;
```

3.1.8 BottomType class

The geo-acoustic properties of the bottom is coded by a single parameter in LYBIN. Bottom types ranging from 1 to 9, where 1 represents a hard, rock type of bottom with low bottom reflection loss, while 9 represents a soft bottom with a high reflection loss. In addition, bottom types 0 and 10 has been added, representing *lossless* and *fully absorbing* bottoms, respectively.

Parameter	Type	Default value	Unit
BottomType	Object (Double[x,3])	(0, 0, 0) (*start, stop, value*)	(Kilometres, Kilometres, -)

Table 3.9 Parameters in the BottomType class.

An example of how BottomType can be used is shown below. In the example two different bottom types are set. From the range of 0 km to 5 km, the bottom type is 4. From the range of 5 km to 10 km, the bottom type is 2.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Bottom type
double[,] bt = new double[2, 3];
bt[0, 0] = 0;
bt[0, 1] = 5;
bt[0, 2] = 4;
bt[1, 0] = 5;
bt[1, 1] = 10;
bt[1, 2] = 2;

Lybin.BottomType = bt;
```

3.1.9 BottomLoss class

Bottom loss is the fraction of energy that is lost after the sound has been reflected from the ocean bottom, usually expressed in dB. The bottom loss is also referred to as *forward scattering* in underwater acoustic terminology. Bottom loss is generally a function of bottom type, gracing angle and frequency. A dataset representing bottom loss is entered into LYBIN in tabular form, giving bottom loss (in dB) for a set of grazing angles. Based on the tabulated values, LYBIN interpolates between tabulated values to create loss values for equidistantly spaced grazing angles.

There is only one parameter in the BottomLoss class, the BottomLossTableCount, given in Table 3.10. The functions in the BottomLoss class are given in Table 3.11.

Parameter	Type	Default value	Unit
BottomLossTableCount *Number of bottom loss tables.*	Integer	1	

Table 3.10 Parameters in the BottomLoss class.

Function	Type	Unit of input parameters
AddBottomLossTable(int start, int stop, object table) *Add another bottom loss table.This function can only be used once the first bottom loss table is added with the SetFirstBottomLossTable function.*	Void	**start**: kilometres, **stop**: kilometres **table**: dB vs. degrees
GetBottomLossTable(int index, out int start, out int stop, out object table) *Get the bottom loss table corresponding to the given index.*	Void	
SetFirstBottomLossTable(int start, int stop, object table) *Set the first bottom loss table.*	Void	

Table 3.11 Functions in the BottomLoss class.

An example of how some of the bottom loss functions can be used is shown below. In the example, the first bottom loss fan is set to be valid from 0 km to 30 km. The loss table consist of the following data: 10 deg , 4.2 dB, 30 deg = 6.4 dB and 80 deg = 9 dB. At the end of the example, the first bottom loss table is fetched back from LYBIN.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Set the first bottom loss table
double[,] bl = new double[3, 2];
bl[0, 0] = 10;
bl[0, 1] = 4.2;
bl[1, 0] = 30;
bl[1, 1] = 6.4;
bl[2, 0] = 80;
bl[2, 1] = 9;
Lybin.SetFirstBottomLossTable(0, 30, bl);

// Get the first bottom loss table
int index = 0;
int start, stop;
object table = new object();
Lybin.GetBottomLossTable(index, out start, out stop, out table);
```

3.1.10 BottomBackScatter class

Bottom back scatter is the fraction of energy that is scattered back towards to the receiver when a ray hits the sea bottom. The bottom back scattering is generally a function of bottom type, grazing angle and frequency. A dataset representing bottom back scattering coefficients is entered into LYBIN in tabular form, giving backscattering coefficients (in dB) for a set of grazing angles. Based on the tabulated values, LYBIN interpolates between tabulated values to create backscattering coefficients for equidistantly spaced grazing angles.

There is only one parameter in the BottomBackScatter class, the BottomBackScatterTableCount, given in Table 3.12. The functions in BottomBackScatter class are given in Table 3.13.

Parameter	Type	Default value	Unit
BottomBackScatterTableCount *Number of bottom back scatter tables.*	Integer	1	

Table 3.12 Parameters in the BottomBackScatter class.

Function	Type	Unit of input parameters
AddBottomBackScatterTable(int start, int stop, object table) *Add another bottom back scatter table. This function can only be used once the first bottom back scatter table is added with the SetFirstBottomBackScatterTable function.*	Void	**start**: kilometres, **stop**: kilometres **table**: dB vs. degrees
GetBottomBackScatterTable(int index, out int start, out int stop, out object table) *Get the bottom back scatter table corresponding to the given index.*	Void	
SetFirstBottomBackScatterTable(int start, int stop, object table) *Set the first bottom back scatter table.*	Void	

Table 3.13 Functions in the BottomBackScatter class.

An example of how the some of the bottom back scatter functions can be used is shown in the code example below. In the example, the first bottom back scatter table is set. At the range from 0 km to 30 km LYBIN shall use the data points: 10 deg = 35 dB, 30 deg = 25 dB and 80 deg = 23 dB. At the end of the example, the first bottom back scatter table is fetched back from LYBIN. For sea surface and bottom reverberation, the back scattering coefficients are given as dB per square meter.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Set the first bottom back scatter table
double[,] bc = new double[3, 2];
bc[0, 0] = 10;
bc[0, 1] = 35;
bc[1, 0] = 30;
bc[1, 1] = 25;
bc[2, 0] = 80;
bc[2, 1] = 23;
Lybin.SetFirstBottomBackScatterTable(0, 30, bc);

// Get the first bottom back scatter table
int index = 0;
int start, stop;
object table = new object();
Lybin.GetBottomBackScatterTable(index, out start, out stop, out table);
```

3.1.11 VolumeBackScatter class

Volume back scatter is fraction of energy scattered back towards the receiver from the sea volume. Scattering elements in the sea volume can be particles or organic life, like plankton, fish or sea mammals. The volume back scatterers are not distributed uniformly in the sea, and can vary considerably as a function of depth, and also on range and time of the day. In LYBIN, the volume back scatter is given as a profile of back scattering coefficients as a function of depth . To find scatter values for the depths not given, linear interpolation is used. The validity of each profile will be given by the corresponding the start range and stop range values.
There is only one parameter in the VolumeBackScatter class, the VolBackScatterProfileCount, given in Table 3.14. The functions in the VolumeBackScatter class are given in Table 3.15.

Parameter	Type	Default value	Unit
VolBackScatterProfileCount *Number of volume back scatter profiles.*	Integer	1	

Table 3.14 Parameters in the VolumeBackScatter class.

Function	Type	Unit of input parameters
AddVolBackScatterProfile(int start, int stop, object profile) *Add another volume back scatter profile.This function can only be used when the first volume back scatter profile is added with the SetFirstVolumeBackScatterFan function.*	Void	**start**: kilometres, **stop**: kilometres **profile**: dB /metre^3
GetVolBackScatterProfile(int index, out int start, out int stop, out object profile) *Get the volume back scatter profile corresponding to the given index.*	Void	
SetFirstVolBackScatterProfile(int start, int stop, object profile) *Set the first volume back scatter profile.*	Void	

Table 3.15 Functions in the VolumeBackScatter class.

An example of how some of the volume back scatter functions can be used is shown below. In the example, the first volume back scatter profile is set. At the range from 0 km to 10 km, LYBIN is to use the values: 10 meters = -80 dB and 50 meters = -92 dB. At the end of the example, the first volume back scatter profile is fetched back from LYBIN.
Volume reverberation back scatter coefficients are given as dB per cubic metre.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Set the first volume back scatter profile
double[,] vc = new double[2, 2];
vc[0, 0] = 10;
vc[0, 1] = -80;
vc[1, 0] = 50;
vc[1, 1] = -92;
Lybin.SetFirstVolBackScatterProfile(0, 10, vc);

// Get the first volume back scatter profile
int index = 0;
int start, stop;
object profile = new object();
Lybin.GetVolBackScatterProfile(index, out start, out stop, out profile);
```

3.1.12 ReverberationAndNoiseMeasurements class

The ReverberationAndNoiseMeasurements class only has one accessible parameter, ReverberationAndNoiseMeasurements, which is listed in Table 3.16. The ReverberationAndNoiseMeasurements can consist of any number of measurements with corresponding ranges. To find values for the ranges not given as measurements, LYBIN uses linear interpolation.

Parameter	Type	Default value	Unit
ReverberationAndNoiseMeasurements *Reverberation and noise measurement as function of range.*	Object (Double[x,2])	(0, 80) (*range, measurement*)	(Metres, dB)

Table 3.16 Parameter in the ReverberationAndNoiseMeasurements class, x can be any number.

An example on how the ReverberationAndNoiseMeasurements can be used is shown below. In the example, two points are inserted. The first is the value 80 dB at a range of 2000 m. The second is the value 70 dB at a range of 8000 m.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

// Reverberation and noise measurements
double[,] ran = new double[2, 2];
ran[0, 0] = 2000;
ran[0, 1] = 80;
ran[1, 0] = 8000;
ran[1, 1] = 70;

Lybin.ReverberationAndNoiseMeasurements = ran;
```

3.1.13 Platform class

The platform class contains all the information about the platform holding the sonar. The platform is most often a ship, but can also be a helicopter or a buoy. The parameters in the platform class are listed in Table 3.17.

Parameter	Type	Default value	Unit
SelfNoise *Noise from the platform that holds the sonar.*	Double	50	dB
SelfNoisePassive *Noise from the platform that holds the sonar. To be used in calculations for passive sonars.*	Double	50	dB
Speed *Speed of the platform that holds the sonar.*	Double	10	Knots

Table 3.17 Parameters in the platform class.

3.1.14 Sensor class

The sensor class contains all the information about the sonar. The parameters in the sensor class are listed in Table 3.18. There are no access functions connected to the sensor class.

Parameter	Type	Default value	Unit
BeamWidthReceiver *Vertical beam width of the receiving part of the sonar.*	Double	15	Degrees
BeamWidthTransmitter *Vertical beam width of the transmitting part of the sonar.*	Double	15	Degrees
Depth *Depth of the sonar.*	Double	5	Meters
DetectionThreshold *The strength of the signal relative to the masking level necessary to see an object with the sonar.*	Double	10	dB
DirectivityIndex *The sonars ability to suppress isotropic noise relative to the response in the steering direction.*	Double	1	dB
Frequency *Centre frequency of the sonar.*	Double	7000	Hz
IntegrationTimePassive *Integration time for the passive sonar.*	Double	1	seconds
PassiveBandWidth *Band width of the passive sonar.*	Double	100	seconds
PassiveFrequency *Centre frequency of the passive sonar.*	Double	800	Hz
SideLobeReceiver *The suppression of the highest side lobe relative to the centre of the beam for the receiving sonar.*	Double	13	dB

Parameter	Type	Default value	Unit
SideLobeTransmitter *The suppression of the highest side lobe relative to the centre of the beam for the transmitting sonar.*	Double	13	dB
SonarTypePassive *Tells whether the passive sonar is broad- or narrowband.* *0: Narrowband* *1: Broadband*	Enumerator	0	
SourceLevel *Source level of the sonar.*	Double	221	dB
SourceLevelPassive *Source level of the possible target in the calculation for passive sonar.*	Double		
SystemLoss *System loss due to special loss mechanisms in the sea or sonar system, not otherwise accounted for.,*	Double	0	dB
TiltReceiver *Tilt of the receiving part of the sonar.*	Double	4	degrees
TiltTransmitter *Tilt of the transmitting part of the sonar.*	Double	4	degrees

Table 3.18 Parameters in the sensor class.

3.1.15 Pulse class

All the information about the pulse is gathered in the pulse class. All the access parameters in the pulse class are listed in Table 3.19 below. The pulse class does not have any access functions.

Parameter	Type	Default value	Unit
EnvelopeFunc *Envelope function of the signal. Currently, only "Hann" is available.*	String	Hann	
FilterBandWidth *Filter bandwidth of the pulse.*	Double	100	Hz
FMBandWidth *Frequency modulation bandwidth of the pulse. Applicable for FM signals only.*	Double	100	Hz
Form *Pulse type:* *FM: Frequency modulated* *CW: Continuous wave*	String	FM	

Parameter	Type	Default value	Unit
Length *Pulse length.*	Double	60	milliseconds
ProsessingGainNoise *Processing gain relative to noise.*	Double	0	dB
ProsessingGainRev *Processing gain relative to reverberation.*	Double	0	dB

Table 3.19 Parameters in the pulse class.

3.2 Initiate calculation

The DoCalculation function initiates a new LYBIN calculation. Before the DoCalculation function is performed, all input parameters must be set, otherwise default parameters are used.

Function	Type
DoCalculation() *Start the calculation.*	Void

Table 3.20 Function for initiation of calculation.

3.3 Calculation results

The calculation results can be accessed through parameters or functions. The result parameters are listed in Table 3.21. Each parameter represents a complete dataset. The result functions give more flexibility in the way that you can access the calculated results. All the functions delivering calculation results are listed in Table 3.22. The formats of the binary output files listed as objects in the tables below are listed in Appendix A. If a calculation fails, the returned value properties will be NULL.

Parameter	Access	Type	Unit
BottomReverberation *Calculated bottom reverberation values.*	Read	Object	dB
ImpulseResponse *Calculated impulse response.*	Read	Object	dB
NoiseAfterProcessing *Calculated noise after processing.*	Read	Double	dB
ProbabilityOfDetection *Calculated probability of detection.*	Read	Object	%
RayTrace *Ray trace from rays transmited within the main lobe of of the transmitting sonar.*	Read	Object	Meters
ResultModelData *The model data used during the calculation.*	Read	String	
ReverberationAndNoiseMeasurment *Total reverberation and noise values given as input to LYBIN, and used in the calculations of signal excess and probability of detection. This parameter will only be availible if reverberation and noise measurements are given to LYBIN before the calculation is performed.*	Read	Object	dB
SignalExcess *Calculated signal excess.*	Read	Object	dB
SurfaceReverberation *Calculated surface reverberation.*	Read	Object	dB
TotalReverberation *Calculated total reverberation.*	Read	Object	dB
TransmissionLossReceiver *Calculated transmission loss from the target to the receiver.*	Read	Object	dB
TransmissionLossTransmitter *Calculated transmission loss from the transmitter to the target.*	Read	Object	dB
TravelTime *Travel time for rays.*	Read	Object	Seconds
VolumeReverberation *Calculated volume reverberation.*	Read	Object	dB

Table 3.21 Parameters containing calculation results.

Function	Type
GetAllResults(out string xmlResult) *Gets all results from the calculation in a single XML-string.*	Void
GetImpulseResponse(out double range, out double depth, out object result) *Get the calculated impulse response*	
GetRay(int index, out object result) *Get the position of ray in range and depth for the ray corresponding to index.*	Void
GetResultModelData(out string xmlData) *Gets all the model data used in the calculation in a single XML-string.*	Void
GetResults(int resultCat, out string xmlResult) *Gets the result specified in resultCat as a XML-string. The possible choices of resultCat are listed in* Table 3.23.	Void
GetResultsBin(int resultCat, out object result) *Gets the result specified in resultCat as an object. The possible choices of resultCat are listed in* Table 3.23.	Void
GetResultsBinValue(int resultCat, int xVal, int yVal, out double result) *Get a single value from the result specified in resultCat and by the indexes x and y. The possible choices of resultCat are listed in* Table 3.23.	Void
GetTravelTime(int index, out object result) *Get the start angle, travel time and position of ray in range and depth for the ray corresponding to index.*	Void

Table 3.22 Functions delivering calculation results

resultCat	Description
0	Transmission loss from transmitter to target
1	Transmission loss from target to receiver
2	Signal excess
3	Probability of detection
4	Total reverberation
5	Surface reverberation
6	Volume reverberation
7	Bottom reverberation
8	Noise after processing

Table 3.23 Available values of resultCat with description.

An example of how some of the result functions can be used is shown below. Three methods are defined, returning the bottom reverberation, the noise after processing and the parameters used in the calculations.

```
LybinCom.LybinModelComBinClass Lybin = new LybinCom.LybinModelComBinClass();

public double[] GetBottomReverberation()
{
   // Initiate the reverberation array
   int NumberOfValues = Lybin.rangeCells;
   double[] BottomReverberationValues = new double[NumberOfValues];

   Object Objekt;
   Lybin.GetResultsBin(7, out Objekt);
   BottomReverberationValues = (double[])Objekt;

   return BottomReverberationValues;
}

public double GetNoise()
{
   double noise;
   Object Objekt;

   Lybin.GetResultsBin(8, out Objekt);
   noise = (double)Objekt;

   return noise;
}

public string GetUsedParameters()
{
   string parameters;
   Lybin.GetResultModelData(out parameters);

   return parameters;
}
```

4 Description of the xml interface

All the parameters in LYBIN can be changed using XML strings, and all the parameters used in and results from the calculations can be retrieved through XML strings. The LYBIN XML interface is described in [5]. The interface described there is still valid. Only changes from the previous documented format are described here.

The structure of the XML strings reflects the class structure in the LYBIN calculation kernel. Every data class shown in Figure 2.4, has its own corresponding XML structure. These XML structures can be entered into LYBIN as single files or together as larger XML files.

For each interface class in LYBIN there is a corresponding hierarchical structure in XML.

4.1 Input data

All the new input parameters are members of the LybinModelData class. The new parameters are listed in Table 4.1. The structure of the XML-file corresponding to the class LybinModelData is shown in Appendix B.1.

Parameter	Type	Default value	Unit
IMPULSERESPONSECALCULATION *Tells LYBIN whether to calculate impulse response or not.* *False: Do not calculate impulse response.* *True: Calculate impulse response.*	Boolean	false	
IMPULSERESPONSEPOSRANGE *Range from the receiver to the point where the impulse response is calculated from.*	Double	100	Meters
IMPULSERESPONSEPOSDEPTH *Depth from the receiver to the point where the impulse response is calculated from.*	Double	100	Meters
TRAVELTIMEANGLERES *The distance in degrees between the start angles of the rays to be used in the travel time calculation.*	Double	1	Degrees

Parameter	Type	Default value	Unit
TRAVELTIMECALCULATION *Tells LYBIN whether to calculate travel time or not.* *False: Do not calculate travel time.* *True: Calculate travel time.*	Boolean	false	
VISULARAYTRACECALCULATION *Tells LYBIN whether to calculate ray trace for visualisation or not.* *False: Do not calculate ray trace for visualisation.* *True: Calculate ray trace for visualisation.*	Boolean	false	
VISUALSURFACEHITS *Number of surface hits alloved in the visual ray trace.*	Integer	2	
VISUALBOTTOMHITS *Number of bottom hits alloved in the visual ray trace.*	Integer	1	
VISUALNUMRAYS *Number of rays in the visual ray trace.*	Integer	50	

Table 4.1 New XML input parameters.

4.2 Calculation results

All the calculation results from LYBIN can be delivered on XML format. With version 5.0 the default unit of all the calculated results is dB. LYBIN 5.0 has three new XML calculation results not described earlier. These are *Ray trace, travel time* and *impulse response*.

All calculation results from LYBIN 5.0 are delivered in dB as default.

4.2.1 Impulse response

The impulse response is calculated for a receiver position given by IMPULSERESPONSEPOSRANGE and IMPULSERESPONSEPOSDEPTH. The impulse response dataset consists of the receiver position, and a set of times and corresponding amplitude values corresponding to the impulse response. Positions are given in meters, time in seconds and relative amplitude (no unit). The structure of the impulse XML output file is given in Appendix B.2.

4.2.2 RayTrace

This ray trace is calculated for the purpose of visualization. Each ray is uniquely defined by the starting angle and a set of numbers giving the ray coordinates (ranges and corresponding depths). The start angle is given in degrees, and the ray coordinates are given in meters. The number of rays in the dataset is given by the parameter VISUALNUMRAYS.
The structure of the ray trace XML output file is given in Appendix B.3.

4.2.3 Travel time

Travel time is calculated for all rays. The distance between the rays is given by the parameter TRAVELTIMEANGLERES. The travel time dataset consists of all the rays calculated for. Each ray has a starting angle and a set of data triplets containing range, depth and travel time. The start angle is given in degrees. The ray ranges and depths are given in meters, and the time in seconds. The structure of the travel time XML output file is given in Appendix B.4.

Appendix A Binary file formats

A.1 Bottom reverberation binary output file

Description	Type	Unit	Remarks
Number of bottom reverberation values	Integer		= N
Bottom reverberation value # 1	Double	dB	....
Bottom reverberation value # 2	Double	dB	....
....	....	....	....
....	....	....	....
....	....	....	....
Bottom reverberation value # N	Double	dB	....

Table A.1 Description of the bottom reverberation binary file format.

A.2 Impulse response binary output file

Description	Type	Unit	Remarks
Number of impulse response values	Integer		= N
Impulse response time value # 1	Double	dB	....
Impulse response value value # 1	Double	dB	....
Impulse response time value # 2	Double	dB	....
Impulse response value value # 2	Double	dB	....
....	....	....	....
....	....	....	....
Impulse response time value # N	Double	dB	....
Impulse response value value # N	Double	dB	....

Table A.2 Description of the impulse response binary output file format.

A.3 Probability of detection binary output file

Description	Type	Unit	Remarks
Number of depths	Integer		= Z
Number of ranges	Integer		= X
Value # 1	Double	%	Depth # 1, range # 1
Value # 2	Double	%	Depth # 1, range # 2
....	....	....	....
....	....	....	....
....	....	%	Depth # 1, range # X-1
....	....	%	Depth # 1, range # X
....	....	%	Depth # 2, range # 1
....	....	%	Depth # 2, range # 2
....	....	....	....
Value # Z*X	Double	%	Depth # Z, range # X

Table A.3 Description of the probability of detection binary file format.

A.4 Ray trace binary output format

Description	Type	Unit	Remarks
Number of rays in calculation	Integer		= N
Number of points in ray # 1	Integer		= M
Start angle of ray # 1	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # 1
Depth of point	Double	Meters	Point # 1 in ray # 1
Range of point	Double	Meters	Point # 2 in ray # 1
Depth of point	Double	Meters	Point # 2 in ray # 1
....	....	....	....
....	....	....	....
Range of point	Double	Meters	Point # M in ray # 1
Depth of point	Double	Meters	Point # M in ray # 1
Number of points in ray # 2	Integer		= P
Start angle of ray # 2	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # 2
Depth of point	Double	Meters	Point # 1 in ray # 2
Range of point	Double	Meters	Point # 2 in ray # 2
Depth of point	Double	Meters	Point # 2 in ray # 2
....	....	....	....
....	....	....	....

Description	Type	Unit	Remarks
Range of point	Double	Meters	Point # P in ray # 2
Depth of point	Double	Meters	Point # P in ray # 2
....	....	....	....
....	....	....	....
Number of points in ray # N	Integer		= Q
Start angle of ray # N	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # N
Depth of point	Double	Meters	Point # 1 in ray # N
Range of point	Double	Meters	Point # 2 in ray # N
Depth of point	Double	Meters	Point # 2 in ray # N
....	....	....	....
....	....	....	....
Range of point	Double	Meters	Point # Q in ray # N
Depth of point	Double	Meters	Point # Q in ray # N

Table A.4 Description of the ray trace binary file format.

A.5 Reverberation and noise measurement binary output format

Description	Type	Unit	Remarks
Number of values	Integer		= N
Value # 1	Double	dB	....
Value # 2	Double	dB	....
....	....	....	....
....	....	....	....
Value # N	Double	dB	....

Table A.5 Description of the reverberation and noise measurement binary file format.

A.6 Signal excess binary output file

Description	Type	Unit	Remarks
Number of depths	Integer		= Z
Number of ranges	Integer		= X
Value # 1	Double	dB	Depth # 1, range # 1
Value # 2	Double	dB	Depth # 1, range # 2
....	....	....	....
....	....	....	....
....	....	dB	Depth # 1, range # X-1
....	....	dB	Depth # 1, range # X
....	....	dB	Depth # 2, range # 1
....	....	dB	Depth # 2, range # 2
....	....	....	....
Value # Z*X	Double	dB	Depth # Z, range # X

Table A.6 Description of the signal excess binary output format.

A.7 Surface reverberation binary output file

Description	Type	Unit	Remarks
Number of surface reverberation values	Integer		= N
Surface reverberation value # 1	Double	dB	
Surface reverberation value # 2	Double	dB	....
....	....	...	....
....	....	...	....
Surface reverberation value # N	Double	dB	....

Table A.7 Description of the surface reverberation binary file format.

A.8 Total reverberation binary output file

Description	Type	Unit	Remarks
Number of total reverberation values	Integer		= N
Total reverberation value # 1	Double	dB	
Total reverberation value # 2	Double	dB	....
....	....	...	....
....	....	...	....
Total reverberation value # N	Double	dB	....

Table A.8 Description of the total reverberation binary file format.

A.9 Transmission loss receiver binary output file

Description	Type	Unit	Remarks
Number of depths	Integer		= Z
Number of ranges	Integer		= X
Value # 1	Double	dB	Depth # 1, range # 1
Value # 2	Double	dB	Depth # 1, range # 2
....	....	....	....
....	....	....	....
....	....	dB	Depth # 1, range # X-1
....	....	dB	Depth # 1, range # X
....	....	dB	Depth # 2, range # 1
....	....	dB	Depth # 2, range # 2
....	....	....	....
Value # Z*X	Double	dB	Depth # Z, range # X

Table A.9 Description of the transmission loss receiver binary output file.

A.10 Transmission loss transmitter binary output file

Description	Type	Unit	Remarks
Number of depths	Integer		= Z
Number of ranges	Integer		= X
Value # 1	Double	dB	Depth # 1, range # 1
Value # 2	Double	dB	Depth # 1, range # 2
....	....	....	....
....	....	....	....
....	....	dB	Depth # 1, range # X-1
....	....	dB	Depth # 1, range # X
....	....	dB	Depth # 2, range # 1
....	....	dB	Depth # 2, range # 2
....	....	....	....
Value # Z*X	Double	dB	Depth # Z, range # X

Table A.10 Description of the transmission loss transmitter binary output file.

A.11 Travel time binary output file

Description	Type	Unit	Remarks
Number of rays in calculation	Integer		= N
Number of points in ray # 1	Integer		= M
Start angle of ray # 1	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # 1
Depth of point	Double	Meters	Point # 1 in ray # 1
Travel time to point	Double	Seconds	Point # 1 in ray # 1
Range of point	Double	Meters	Point # 2 in ray # 1
Depth of point	Double	Meters	Point # 2 in ray # 1
Travel time to point	Double	Seconds	Point # 2 in ray # 1
....	....	....	....
....	....	....	....
Range of point	Double	Meters	Point # M in ray # 1
Depth of point	Double	Meters	Point # M in ray # 1
Travel time to point	Double	Seconds	Point # M in ray # 1
Number of points in ray # 2	Integer		= P
Start angle of ray # 2	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # 2
Depth of point	Double	Meters	Point # 1 in ray # 2
Travel time to point	Double	Seconds	Point # 1 in ray # 2
Range of point	Double	Meters	Point # 2 in ray # 2
Depth of point	Double	Meters	Point # 2 in ray # 2
Travel time to point	Double	Seconds	Point # 2 in ray # 2
....	....	....	....
....	....	....	....
Range of point	Double	Meters	Point # P in ray # 2
Depth of point	Double	Meters	Point # P in ray # 2
Travel time to point	Double	Seconds	Point # P in ray # 2
....	....	....	....
....	....	....	....
Number of points in ray # N	Integer		= Q
Start angle of ray # N	Double	Degrees	
Range of point	Double	Meters	Point # 1 in ray # N
Depth of point	Double	Meters	Point # 1 in ray # N
Travel time to point	Double	Seconds	Point # 1 in ray # N
Range of point	Double	Meters	Point # 2 in ray # N
Depth of point	Double	Meters	Point # 2 in ray # N
Travel time to point	Double	Seconds	Point # 2 in ray # N

Description	Type	Unit	Remarks
....	....	....	....
....	....	....	....
Range of point	Double	Meters	Point # Q in ray # N
Depth of point	Double	Meters	Point # Q in ray # N
Travel time to point	Double	Seconds	Point # Q in ray # N

Table A.11 Description of the travel time binary file format.

A.12 Volume reverberation binary output file

Description	Type	Unit	Remarks
Number of volume reverberation values	Integer		= N
Volume reverberation value # 1	Double	dB	
Volume reverberation value # 2	Double	dB	....
....	....	....	....
....	....	....	....
Volume reverberation value # N	Double	dB	....

Table A.12 Description of the volume reverberation binary file format.

Appendix B XML file formats

B.1 XML input file to the LybinModelData class

```xml
<?xml version="1.0" ?>
<MODELFILE>
     <FILEINFO>
          <FORMATVERSION>2.0</FORMATVERSION>
     </FILEINFO>
     <MODELPARAMETERS>
          <MAXRANGE>10000</MAXRANGE>
          <RANGECELLSIZE>200</RANGECELLSIZE>
          <RANGESTEPS>500</RANGESTEPS>
          <RANGECELLS>50</RANGECELLS>
          <MAXDEPTH>100</MAXDEPTH>
          <DEPTHCELLSIZE>2</DEPTHCELLSIZE>
          <DEPTHSTEPS>1000</DEPTHSTEPS>
          <DEPTHCELLS>50</DEPTHCELLS>
          <TRLRAYS>1000</TRLRAYS>
          <MAXBORDERHITS>5000</MAXBORDERHITS>
          <TERMINATIONINTENSITY>1e-016</TERMINATIONINTENSITY>
          <SIGNALEXCESSCONSTANT>3</SIGNALEXCESSCONSTANT>
          <DOPASSIVECALCULATION>false</DOPASSIVECALCULATION>
          <REVNOISECALCULATION>0</REVNOISECALCULATION>
          <USEMEASUREDBOTTOMLOSS>false</USEMEASUREDBOTTOMLOSS>
          <IMPULSERESPONSECALCULATION>false</IMPULSERESPONSECALCULATION>
          <IMPULSERESPONSEPOSRANGE>100</IMPULSERESPONSEPOSRANGE>
          <IMPULSERESPONSEPOSDEPTH>100</IMPULSERESPONSEPOSDEPTH>
          <TRAVELTIMEANGLERES>1</TRAVELTIMEANGLERES>
          <TRAVELTIMECALCULATION>false</TRAVELTIMECALCULATION>
          <VISUALRAYTRACECALCULATION>false</VISUALRAYTRACECALCULATION>
          <VISUALSURFACEHITS>2</VISUALSURFACEHITS>
          <VISUALBOTTOMHITS>1</VISUALBOTTOMHITS>
          <VISUALRAYS>false</VISUALRAYS>
     </MODELPARAMETERS>
</MODELFILE>
```

B.2 Impulse response XML output file

```xml
<?xml version="1.0"?>
<LYBINRESULTS>
     <IMPULSERESPONSE>
          <RANGE>15</RANGE>
          <DEPTH>15</DEPTH>
          <SIGNAL>
               <TIME>1.00</TIME>
               <VALUE>50.00</VALUE>
               <TIME>1.20</TIME>
               <VALUE>52.00</VALUE>
                    ...
               <TIME>8.00</TIME>
               <VALUE>4.00</VALUE>
          </SIGNAL>
     </IMPULSERESPONSE>
</LYBINRESULTS>
```

B.3 Ray trace XML output file

```xml
<?xml version="1.0"?>
<LYBINRESULTS>
     <RAYTRACE>
          <RAY>
               <STARTANGLE>15</STARTANGLE>
               <POINT>
                    <RANGE>1.00</RANGE>
                    <DEPTH>50.00</DEPTH>
               </POINT>
               <POINT>
                    <RANGE>6.00</RANGE>
                    <DEPTH>60.00</DEPTH>
               </POINT>
                    ...
               <POINT>
                    <RANGE>6000.00</RANGE>
                    <DEPTH>20.00</DEPTH>
               </POINT>
          </RAY>
          <RAY>
               ...
          </RAY>
     </RAYTRACE>
</LYBINRESULTS>
```

B.4 Travel time XML output file

```
<?xml version="1.0"?>
<LYBINRESULTS>
     <TRAVELTIME>
          <RAY>
               <STARTANGLE>15</STARTANGLE>
               <POINT>
                    <RANGE>1.00</RANGE>
                    <DEPTH>50.00</DEPTH>
                    <TIME>0.000667</TIME>
               </POINT>
               <POINT>
                    <RANGE>6.00</RANGE>
                    <DEPTH>60.00</DEPTH>
                    <TIME>0.004000</TIME>
               </POINT>
                    ...
               <POINT>
                    <RANGE>6000.00</RANGE>
                    <DEPTH>20.00</DEPTH>
                    <TIME>6.000000</TIME>
               </POINT>
          </RAY>
          <RAY>
              ...
          </RAY>
     </TRAVELTIME>
</LYBINRESULTS>
```

References

[1] S Mjølsnes (2000): *LYBIN SGP-180(C) – Model Description*, The Royal Norwegian Navy Materiel Command, Bergen

[2] Teleplan GLOBE, http://www.teleplanglobe.com/index.php?page=maria

[3] A Gjersøe and F Hermansen (2008): *Simson Fennikel – Design dokument for Maria add-in for utvelgelse og visning av data på kart*, Norwegian Defence Research Institute, FFI/RAPPORT 2008/02182

[4] M Bosseng and A Gjersøe (2008): *Simson Fennikel – User Manual*, Norwegian Defence Research Institute, FFI/RAPPORT 2008/02183

[5] E Dombestein and S Alsterberg (2006): *LYBIN XML grensesnitt versjon 1*, Norwegian Defence Research Institute, FFI/RAPPORT 2006/00266

Abbreviations

FFI	Norwegian Defense Research Institute
NDLO	Norwegian Defense Logistic Organization
GFA	Government Furnished Assets
LYBIN	Lydbane og intensitetsprogram (acoustic model)
XML	Extensible markup language