# ISB Publication

## Business Data Architecture Data Types

**Doc Ref:** ISB-000076  
**Version:** 7.0  
**Issue Date:** 24/06/2014

### Document Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Issue Date</th>
<th>Modified by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-0</td>
<td>Final</td>
<td>September 2009</td>
<td>Nirmal Laddha</td>
<td>New document</td>
</tr>
<tr>
<td>2-0</td>
<td>Final</td>
<td>23/03/2010</td>
<td>Nirmal Laddha</td>
<td>Updated with new data types required to address issue of signed integer and characters from European languages.</td>
</tr>
<tr>
<td>3-0</td>
<td>Final</td>
<td>28/03/2011</td>
<td>Steve Palmer</td>
<td>Addition of Email address to include Validating Regular Expression. Addition of Base64Binary for the transfer via XML of image data</td>
</tr>
<tr>
<td>4-0</td>
<td>Final</td>
<td>20/09/2013</td>
<td>Martyn Grandy</td>
<td>Transferred to new ISB format + editorial changes.</td>
</tr>
<tr>
<td>5-0</td>
<td>Final</td>
<td>01/05/2014</td>
<td>Steve Palmer</td>
<td>Issued for internal release</td>
</tr>
<tr>
<td>6-0</td>
<td>Final</td>
<td>28/04/2014</td>
<td>Steve Palmer</td>
<td>Published version for review</td>
</tr>
<tr>
<td>7-0</td>
<td>Final</td>
<td>24/06/2014</td>
<td>Steve Palmer</td>
<td>Revisions following external review and ISB Architecture SIG review</td>
</tr>
</tbody>
</table>

© Crown copyright 2014
## Contents

1. **Introduction** ................................................................. 3  
2. **Requirements of the ISB Business Data Architecture** ....................................................... 3  
3. **Principles** ................................................................. 4  
   3.1 Data Types ............................................................... 4  
   3.2 Reference Data Validation ............................................. 4  
   3.3 Data Types used by the ISB Business Data Architecture .................................................. 5  
   3.4 Logical vs Physical Primitive Data Types .................................................. 6  
4. **ESCS Logical Primitive Data Types** ................................... 8  
5. **XML Schema Primitive Data Types** .................................. 11  

### Appendix A: Notes .......................................................... 13  
1. Decimal notation ............................................................. 13  
2. ASCII, Unicode and UTF-8 .................................................. 13  
3. base64Binary ............................................................... 14  
4. Primitive/Base Data Types .................................................. 14  
5. Enumerated Data Types ..................................................... 15  

### Appendix B: Other Physical Encodings ................................ 16  

### Appendix C: Version History ............................................. 18  
Changes from Version 4.0 ..................................................... 18  

**Copyright Notice** .......................................................... 19
1 INTRODUCTION

This document provides a definition of the data types in the ESCS ISB Business Data Architecture, Business Data Standards and Technical Data Standards.

The ISB Business Data Architecture and Business Data Standards identify logical data designs that can be used as the basis of:

- a system design,
- an operational data store,
- data exchange standards.

Details can be found in:


2 REQUIREMENTS OF THE ISB BUSINESS DATA ARCHITECTURE

The methodology used by the ISB for development of the Business Data Standards is based on Entity Relationship modelling (for further information on the methodology used please refer to the document BDA-Data-Architecture-Standards available from the ISB Secretariat and due to be published on the ISB web site). Entity Relationship Modelling covers the design of entities and attributes. Attributes in an Entity Relationship Diagram (ERD) require certain meta data be assigned to them and one of the meta data items is the data type. The objective of this document is to

1) identify the data types required by the ERD model for use when developing Business Data Standards
2) identify the meta data available for each data type
3) identify the difference between the data types used for the logical Business Data Standards and those that would be used for a specific encoding
4) provide a number of examples of encoding specific translations of the ERD data types
3 PRINCIPLES

3.1 Data Types

Within any data design, it is necessary to be specific about the nature of information so that all users can interpret data in a consistent way. Data stored in a computer is held as a string of bits (0 or 1). However, the value 0011001000110000 can mean either the ASCII characters “20” or the decimal number 12848. Computers interpret the string of bits depending upon its context. Context is made specific in any programming language by specifying the data type.

There are commonly two categories of data type (see 6.4 below):

1) Primitive data types – eg Integer, Character

2) Object (or user defined class) data types – eg, enumeration

To determine whether we have a primitive data type or object (user class) data type we can use the following acid test – “Can I assign an in stream value to the data type in a program?” ie variable_name=1. If the variable_name has been defined with a primitive data type such as an integer data type then we can. However, if we wanted to use a data type of enumerated we first have to create a new object data type class. If we wanted to define an enumerated data type of FRUIT with the restricted values of APPLE, ORANGE or GRAPE we first have to define the scope of the enumerated data type and its underlying primitive data type. Thus if we say that APPLE=0, ORANGE=1 and GRAPE =2, we have assigned the primitive data type to be an integer and so the possible values of FRUIT are cast to that primitive data type. We can now use the new object data type of FRUIT within our program and it will have only the values of APPLE, ORANGE and GRAPE available to it. We cannot, as in our acid test above, assign a new value of PEACH to FRUIT by simply stating FRUIT=PEACH. Therefore, an enumerated data type is not a primitive data type. For further information on enumerated data types see section 6.5

3.2 Reference Data Validation

When defining a data architecture the application of validation rules to reference data attributes requires a specific methodology approach for the following reasons:

1. Validation rules are application and context specific. An example is age. If the age of a person was restricted to the range 1-99 and the particular contextual use of a person was school children, the restriction to a maximum of 99 would be excessive
2. Validation rules can change. Again for the above age example, a specific policy area may consider a child as being a specific age range. However, if there is a policy change then that age range may be adjusted.

3. In a data exchange environment, different systems may have different requirements for validation. A school MIS system may only hold information around children whereas an Operational Data Store could hold data around children, workforce and organisations. The school MIS may need require a certain level of validation before it will accept the information if it is involved in a business 2 business exchange, whereas the ODS is interested in just reflecting the data in the enterprise and so will have a much lower validation requirement.

4. The resulting data block schema design supports many data exchanges through a common schema. As each use of the same data block can be for various contextual purposes, and we don’t want to impact the data exchange interfaces when a validation range changes, data types are not enumerated in the schema.

To support the above principles, the Business Data Architecture does not constrain Business Data Standards attribute data types with range based validation rules (ie Learner age not <3 and not > 20). It instead defines the common enterprise data that applications should ideally use when needed through the provision of the size of ranges (age field length is Integer 999) for implementers considering transfer and storage capabilities and this ensures interoperability.

The actual validation ranges for an application to apply to a reference type entity are provided through the provision of a Controlled List. Each Controlled List defines a specific list of values that a logical data item may take. The way these values are represented for data exchange is defined in the relevant Controlled List and Technical Data Standards.

Consequently, the strategy and principles for applying range validation rules is the responsibility of each application.

3.3 Data Types used by the ISB Business Data Architecture

The ISB Business Data Architecture decouples contextual use and validation ranges from the data model attributes for the reasons explained in sections 3.2. Therefore, the Business Data Standards only require the assignment of a primitive data type to each attribute. Object or user class data types would be assigned as required for the specific encoding, program language or solution as the methods used can vary. As the intention of this document is to support the Business Data Standards ERD model data type requirements then only primitive data types are considered.
3.4 Logical vs Physical Primitive Data Types

Whilst all computer languages and databases use primitive data types they vary slightly in the names they use for the same primitive data type concept. As such there is a need to define these data types in a manner that is free from any specific tool, but in a way that can readily be realised within any tool. This implementation-independent design is called a “logical” representation. Implementation-specific designs are called “physical” representations.

We define a number of primitive logical data types in this document. We also provide examples of a number of physical representations of these logical data types. This is illustrated as follows:

It is expected that most data will be interchanged using XML, so this document includes a standard conversion between the logical data types and XML schema data types used in the set of related Technical Data Standards provided by ISB. Implementers are required to conform to these Technical Data Standards to ensure the correct exchange of data between systems.

Other physical data representations, such as CSV, are possible but are not generally recommended as there are no standards for CSV or agreed data type declarations and so could produce data conversion problems when used across different systems or tools. Where CSV is a supported representation of data, encoding details will be specified in the relevant Technical Data Standard (TDS).
Examples of physical representations of data types are also provided as appendices for Microsoft SQL Server and Oracle database systems. These are provided for illustration only to aid implementers in choosing the data types to use. See Appendix B. Other physical data representations may be added by ISB if they are required in the future.
4 ESCS LOGICAL PRIMITIVE DATA TYPES

The following table describes the labels and definitions for those Logical Primitive Data Types used in the Business Data Standards. When compiling a list of logical data types to use, it was observed that XML refers to Simple Elements as being the base data item, Java refers to data types as Simple_Integer, Simple.Decimal, etc. Oracle also uses the Simple_Integer data type. Therefore, the following table uses this standard by applying the prefix of “Simple” to the commonly used data types. Only the Unicode String data type is exempt from this as to reflect this data type in XML it uses the Simple_String with the application of the UTF8 character set encoding.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Parameters</th>
<th>Type Validation</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Simple_String (a,b)| A string of characters each selected from the basic ASCII character set. (see Appendix A). | a = minimum length  
b = maximum length | Within length constraints.  
Basic ASCII character set only. | May be used for elements with constrained value ranges, eg element ID |
| Unicode_String (a,b)| A string of characters each selected from the Unicode Character Database. (see Appendix A). | a = minimum length  
b = maximum length | Within length constraints. | For general purpose text fields and user entered information          |
| Simple_Integer (a,b)| A signed numeric whole number value.                                        | a = minimum digits  
b = maximum digits | Within data type range.     |                                                                       |
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Parameters</th>
<th>Type Validation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple_Decimal (p,s)</td>
<td>Signed decimal value with precision p and scale s. For example, a monetary</td>
<td>p = maximum number of digits</td>
<td>Within precision and scale.</td>
<td>Eg Simple_Decimal(7,2) = 99999.99</td>
</tr>
<tr>
<td></td>
<td>value limited to 2 decimal places.</td>
<td>s = maximum number of digits to the right of decimal point</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple_Flag</td>
<td>Boolean.</td>
<td></td>
<td>Allowed values only.</td>
<td>The recommended values are “Y” or “N”. Due to this restriction, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>actual data types used in encodings will be suitable for handling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>character data of the values “Y” or “N” only</td>
</tr>
<tr>
<td>Simple_Date</td>
<td>Date information containing year, month and day.</td>
<td></td>
<td>A valid date.</td>
<td></td>
</tr>
<tr>
<td>Simple_Time</td>
<td>Time information containing hour(s), minute(s) and second(s). May include</td>
<td></td>
<td>A valid time using a 24-hour clock</td>
<td>Midnight is 0 hours, 0 minutes, 0 seconds.</td>
</tr>
<tr>
<td></td>
<td>fractions of a second.</td>
<td></td>
<td>within a specified time zone.</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>Description</td>
<td>Parameters</td>
<td>Type Validation</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------</td>
<td>-----------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Simple_Date_Time</td>
<td>Date and time information together.</td>
<td></td>
<td>See above.</td>
<td></td>
</tr>
<tr>
<td>Simple_Binary (a,b)</td>
<td>Object stored as encoded binary data. For example, a picture.</td>
<td>a = minimum length</td>
<td>Within length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b = maximum length</td>
<td>constraints.</td>
<td></td>
</tr>
</tbody>
</table>
5 XML SCHEMA PRIMITIVE DATA TYPES

This section maps the logical primitive data types to XML schema primitive data types for data exchange. These mappings apply consistently across all XML Technical Data Standards produced by ECSC ISB.

<table>
<thead>
<tr>
<th>Logical Data Type</th>
<th>XML Schema Data Type</th>
<th>Constraints</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple_String (a,b)</td>
<td>string</td>
<td>minLength = a</td>
<td><a href="http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string">http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxLength = b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Basic ASCII character set only</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See Appendix A for character set details</td>
</tr>
<tr>
<td>Unicode_String (a,b)</td>
<td>string</td>
<td>minLength = a</td>
<td><a href="http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string">http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxLength = b</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UTF-8 encoding is used</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See Appendix A for character set details</td>
</tr>
<tr>
<td>Simple_Integer (a,b)</td>
<td>integer</td>
<td>(a) parameter – no equivalent in xml</td>
<td><a href="http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#integer">http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#integer</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalDigits = b</td>
<td></td>
</tr>
<tr>
<td>Simple_Decimal (p,s)</td>
<td>decimal</td>
<td>totalDigits = p</td>
<td><a href="http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#decimal">http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#decimal</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fractionDigits = s</td>
<td></td>
</tr>
<tr>
<td>Simple_Flag</td>
<td>string</td>
<td>pattern = [YN]{1}</td>
<td><a href="http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string">http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#string</a></td>
</tr>
<tr>
<td>Simple_Date</td>
<td>date</td>
<td>CCYY-MM-DD</td>
<td><a href="http://www.w3.org/TR/NOTE-datetime">http://www.w3.org/TR/NOTE-datetime</a></td>
</tr>
<tr>
<td>Logical Data Type</td>
<td>XML Schema Data Type</td>
<td>Constraints</td>
<td>Reference</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Simple_Time          | time                 | hh:mm:ss(.sss)[Z][(+|-)hh:mm] seconds and additional fractions of time such as hundredths of seconds are optional | http://www.w3.org/TR/NOTE-datetime  
| Simple_Date_Time     | dateTime             | CCYY-MM-DDThh:mm:ss(.sss)[Z][(+|-)hh:mm] seconds and additional fractions of time such as hundredths of seconds are optional | http://www.w3.org/TR/NOTE-datetime  
| Simple_Binary(a,b)   | base64Binary         | minLength = a  
maxLength = b                                                            | http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/#base64Binary |
APPENDIX A: NOTES

1. Decimal notation

“p” specifies the maximum total number of decimal digits available, both to the left and to the right of the decimal point. The precision must be a value from 1 through to the maximum precision.

“s” specifies the maximum number of decimal digits available to the right of the decimal point. Scale must be a value from 0 through to p. Scale can be specified only if precision is specified. The default scale is 0; therefore, 0 <= s <= p.

2. ASCII, Unicode and UTF-8

Character standards are needed to define consistent representations and manipulation of text expressed or used in writing systems.

The Unicode character standard lists the characters from the alphabets used around the world (including Latin, Cyrillic, Farsi etc etc) within the Unicode Character Database. This specifies each character from each alphabet and assigns to each a unique number. Unicode standards are maintained by the Unicode Consortium, a non-profit organisation that coordinates Unicode’s development. See [http://www.unicode.org/](http://www.unicode.org/).

ESCS ISB has approved data types that can contain the basic ASCII character set (referred to as Basic Latin (ASCII)) and the extended Latin Unicode character set. The approved list is based on the [XML 1.0 character range](http://www.w3.org/TR/REC-xml/#charsets) (Section 2.2 Characters). This includes:

- U+0009, U+000A, U+000D
- U+0020–U+D7FF, U+E000–U+FFFD: all surrogates, U+FFE0 and U+FFFF are forbidden;
- U+10000–U+10FFFF: this includes all code points in supplementary planes, including non-characters.

The following controls are only valid in certain contexts in XML 1.0 documents, and therefore prohibited

- U+007F–U+0084, U+0086–U+009F: this includes a C0 control character and all but one C1 control.

To see how to use special characters refer to [W3Schools.com – Characters](http://www.w3schools.com/characters)

The following are the references to the code pages that cover the above. When referring to the following always observe the restrictions above:
1. **Basic Latin (ASCII)**: page with code points U+0000 to U+007F
2. **Latin-1 Supplement**: page with code point U+0080 to U+00FF
3. **Latin Extended A**: page with code points U+0100 to U+017F
4. **Latin Extended B**: page with code points U+0180 to U+024F
5. **Latin Extended C**: page with code points U+2C60 to U+2C7F
6. **Latin Extended D**: page with code points U+A720 to U+A7FF
7. **Latin Extended Additional**: page with code points U+1E00 to U+1EFF

The Unicode Consortium also recommends use of the UTF-8 encoding for the physical encoding of the Unicode Character Database. Support for the transfer of UTF-8 (Unicode Transformation Format) is required within a physical TDS. This is a variable length encoding of the full Unicode character set whilst also giving an efficient single byte encoding for the 128 ASCII characters. UTF-8 is described in [RFC 3629](https://tools.ietf.org/html/rfc3629).

3. **base64Binary**

base64Binary is a method used to encode binary information into a simple character string. It is used for transferring images, video, audio or other media within an XML document. A separate element is required that identifies the format so the data can be decoded, eg JPEG, GIF etc.

base64Binary is described in [RFC 2045](https://tools.ietf.org/html/rfc2045) and [XML Schema Second Edition 2004](https://www.w3.org/TR/xmlschema-2/).

4. **Primitive/Base Data Types**

References:

- [https://wiki.openoffice.org/wiki/Base/Data_Types](https://wiki.openoffice.org/wiki/Base/Data_Types)
External research:

- [http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html](http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html)
- [http://www.tutorialspoint.com/java/java_basic_datatypes.htm](http://www.tutorialspoint.com/java/java_basic_datatypes.htm)

### 5. Enumerated Data Types

As described in section 3.1 enumerated data types are a cast to a primitive data type. In xml this can be seen clearly in the following extracted from the ISB xsd schema:

```xml
<xs:simpleType name="Flag_DataType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Y"/>
    <xs:enumeration value="N"/>
  </xs:restriction>
</xs:simpleType>
```

The above demonstrates that the enumerated data type of Flag_DataType has a primitive data type defined by base="xs:string"

Enumerated data types are, however, used within the ISB schema where appropriate. They are not used for Controlled Lists for the reasons stated above in section 3.2. However, where there is a need to constrain a Business Data Standard attribute to a set of finite values that affect the behaviour of the data exchange, then a constrained enumerated element is appropriate. The ISB schema defines a section of the schema titled “Global Data Types”. This section currently contains one enumerated data type that defines a Flag_DataType as a cast to the Logical Simple_Flag Primitive Data Type and constrains it to the values of “Y” or “N”. This is required as these values must never change as they control the behaviour of the data exchange.

As this document concerns itself with defining only primitive data types for the reasons explained in section 3.1 then enumerated data types are not included in the above list of logical primitive data types.

References:


External research:

APPENDIX B: OTHER PHYSICAL ENCODINGS

The following table identifies two encodings that are using the ESCS Data Architecture and provides a list of equivalent data types within the encoding. This is included to assist with understanding how data could be stored so as to allow faithful import and export of data via an ISB TDS.

<table>
<thead>
<tr>
<th>ESCS Logical Data Type</th>
<th>Sql Server 2008/2012 Data Types/Notes</th>
<th>Oracle 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple_String (a,b)</td>
<td>varchar (b</td>
<td>max)</td>
</tr>
<tr>
<td></td>
<td>b = max length</td>
<td>b = max length</td>
</tr>
<tr>
<td></td>
<td>max is for long strings over 8,000 bytes</td>
<td></td>
</tr>
<tr>
<td>Unicode_String (a,b)</td>
<td>nvarchar (b</td>
<td>max)</td>
</tr>
<tr>
<td></td>
<td>b = max length</td>
<td>b = max length</td>
</tr>
<tr>
<td></td>
<td>max is for long strings over 4,000 bytes</td>
<td></td>
</tr>
<tr>
<td>Simple_Integer (a,b)</td>
<td>smallint, int or big int as appropriate to the value range required</td>
<td>number (b)</td>
</tr>
<tr>
<td></td>
<td>b = max number of digits</td>
<td>b = max number of digits</td>
</tr>
<tr>
<td>Data Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Simple_Decimal (p,s)   | decimal (p,s)  
                      p = precision. The maximum number of digits  
                      s = scale. The maximum number of digits to the right of the decimal point  |
| Simple_Flag            | Either bit or char(1)  
                      If bit is used then the data should be exchanged or displayed as either Y (bit value = 1) or N (bit value – 0)  
                      If char(1) is used then it should store Y or N                           |
| Simple_Date            | date                                                                      |
| Simple_Time            | datetimeoffset to include the time zone with a default date such as 0001-01-01 |
| Simple_Date_Time       | See above                                                                |
| Simple_Binary(a,b)     | varbinary (b | max) or Remote Blob Storage (RBS)  
                      b = max length  
                      max is for long objects over 8,000 bytes                                |
| Simple_Binary          | blob                                                                     |
APPENDIX C: VERSION HISTORY

This section explains the changes that have been made to previous major versions of this document. Details of changes are not usually carried forward between major versions of a document. Please refer to previous versions.

Changes from Version 4.0

1) Review and rationalisation of logical data types
2) Added requirement to support UTF-8
3) Removed validating regular expression that restricted the use of the Unicode character set
4) Addition of standard XML schema data types to be used across XML TDS for data exchange
5) Mapping between the logical data types and XML Schema Data Types Second Edition 2004
6) Added diagram to introduction and added some supporting text
7) Added Principles section
8) Added notes on Controlled Lists
9) Added notes on validation rules
10) Updated Appendix B
11) Added new section 2, updated sections 3, 5 and 6.1.2 and added new sections 6.1.4 and 6.1.5
COPYRIGHT NOTICE

© Crown copyright 2014

The Information Standards Board (ISB) is an advisory body to the Department for Education (DfE) and the Department for Business, Innovation and Skills (BIS). The information it produces is subject to Crown copyright, which is administered by the National Archives.

The Crown copyright protected information in this document (other than ISB or Departmental logos) may be reproduced free of charge in any format or medium under the terms of the Open Government Licence, available from the National Archives website.

Any reuse is subject to the material being reproduced accurately and not used in a misleading context. It must be acknowledged as being protected by Crown copyright and the title of the source material must be supplied with the ISB named as the corporate author.

Authorisation to reproduce any information which is identified as being the copyright of a third party must be obtained from the copyright holders concerned.