

Australian/New Zealand Standard™

Geographic information—Encoding



AS/NZS ISO 19118:2012

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-004, Geographical Information/Geometries. It was approved on behalf of the Council of Standards Australia on 23 August 2012 and on behalf of the Council of Standards New Zealand on 10 August 2012.
This Standard was published on 5 October 2012.

The following are represented on Committee IT-004:

ANZLIC—the Spatial Information Council
Australasian Fire and Emergency Service Authorities Council
Australian Antarctic Division
Australian Hydrographic Office
Australian Map Circle
CSIRO Exploration and Mining
Department of Lands, NSW
Department of Primary Industries and Water Tasmania
Land Information New Zealand
Office of Spatial Data Management
Geoscience Australia
Mercury Project Solutions
The University of Melbourne

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.saiglobal.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR AS/NZS ISO 19118.

Australian/New Zealand Standard™

Geographic information—Encoding

First published as AS/NZS ISO 19118:2006.
Second edition 2012.

COPYRIGHT

© Standards Australia Limited/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968 (Australia) or the Copyright Act 1994 (New Zealand).

Jointly published by SAI Global Limited under licence from Standards Australia Limited, GPO Box 476, Sydney, NSW 2001 and by Standards New Zealand, Private Bag 2439, Wellington 6140.

ISBN 978 1 74342 232 8

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-004, Geographical Information/Geometrics, to supersede AS/NZS ISO 19118:2006, *Geographic information—Encoding*.

The objective of this Standard is to provide information system designers and analysts with the requirements for defining encoding rules to be used for interchange of geographic data within the ISO 19100 series of International Standards, including requirements for creating encoding rules based on UML schemas, requirements for creating encoding services, and an informative XML-based encoding rule for neutral interchange of geographic data.

This Standard is identical with, and has been reproduced from, ISO 19118:2011, *Geographic information—Encoding*.

As this Standard is reproduced from an International Standard, the following applies:

- (a) Its number appears on the cover and title page while the International Standard number appears only on the cover.
- (b) In the source text ‘this International Standard’ should read ‘this Australian/New Zealand Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO/TS		AS/NZS ISO	
19103	Geographic information—Conceptual schema language	19103	Geographic information—Conceptual schema language
ISO			
19109	Geographic information—Rules for application schema	19109	Geographic information—Rules for application schema
		AS ISO	
8601	Data elements and interchange formats—Information interchange—Representation of dates and times	8601	Data elements and interchange formats—Information interchange—Representation of dates and times

The terms ‘normative’ and ‘informative’ have been used in this Standard to define the application of the appendix to which they apply. A ‘normative’ appendix is an integral part of a Standard, whereas an ‘informative’ appendix is only for information and guidance.

CONTENTS

1	Scope	1
2	Conformance	1
2.1	Introduction	1
2.2	Conformance classes related to encoding rules	1
2.3	Conformance classes related to encoding services	1
3	Normative references	2
4	Terms and definitions	2
5	Symbols and abbreviated terms	6
6	Fundamental concepts and assumptions	7
6.1	Concepts	7
6.2	Data interchange	7
6.3	Application schema	8
6.4	Encoding rule	9
6.5	Encoding service	10
6.6	Transfer service	10
7	Character repertoire	11
8	Generic instance model	11
8.1	Introduction	11
8.2	Relation between UML and the instance model	14
9	Encoding rules	14
9.1	Introduction	14
9.2	General encoding requirements	15
9.3	Input data structure	17
9.4	Output data structure	17
9.5	Conversion rules	18
9.6	Examples	18
10	Encoding service	18
Annex A (normative)	XML-based encoding rule	20
Annex B (normative)	Abstract test suit	21
Annex C (informative)	XML-based encoding rule in use by communities	25
Bibliography		68

INTRODUCTION

This International Standard specifies the requirements for defining encoding rules used for interchange of geographic data within the set of International Standards known as the “ISO 19100 series”. An encoding rule allows geographic information defined by application schemas and standardized schemas to be coded into a system-independent data structure suitable for transport and storage. The encoding rule specifies the types of data being coded and the syntax, structure and coding schemes used in the resulting data structure. The resulting data structure can be stored on digital media or transferred using transfer protocols. It is intended that the data be read and interpreted by computers, but data can be in a form that is human readable.

The choice of one encoding rule for application-independent data interchange does not exclude application domains and individual nations from defining and using their own encoding rules that can be platform dependent or more effective with regard to data size or processing complexity. XML is a subset of ISO/IEC 8879 and has been chosen because it is independent of computing platform and interoperable with the World Wide Web.

This International Standard is divided into three logical sections. The requirements for creating encoding rules based on UML schemas are specified in Clauses 6 to 9. The requirements for creating encoding service are specified in Clause 10, and the requirements for XML-based encoding rules are specified in Annex A.

The XML-based encoding rule is intended for use as a neutral data interchange. It relies on the Extensible Markup Language (XML) and the ISO/IEC 10646 character set standards.

The geographic information standards are organized within the set of International Standards known as the “ISO 19100 series”. The background and the overall structure of this series of International Standards and the fundamental description techniques are defined in ISO 19101, ISO/TS 19103 and ISO/TS 19104.

Users of this International Standard can develop application schemas to formally describe geographic information. An application schema is compiled by integrating elements from other standardized conceptual schemas (e.g. ISO 19107). How this integration takes place is described in ISO 19109. The set of International Standards known as the “ISO 19100 series” also defines a set of common services that are available when developing geographic information applications. The common services are generally defined in ISO 19119 and cover access to, and processing of, geographic information according to the common information model. This International Standard covers implementation issues.

AUSTRALIAN/NEW ZEALAND STANDARD

Geographic information—Encoding**1 Scope**

This International Standard specifies the requirements for defining encoding rules for use for the interchange of data that conform to the geographic information in the set of International Standards known as the “ISO 19100 series”.

This International Standard specifies

- requirements for creating encoding rules based on UML schemas,
- requirements for creating encoding services, and
- requirements for XML-based encoding rules for neutral interchange of data.

This International Standard does not specify any digital media, does not define any transfer services or transfer protocols, nor does it specify how to encode inline large images.

2 Conformance**2.1 Introduction**

Two sets of conformance classes are defined for this International Standard.

2.2 Conformance classes related to encoding rules

All encoding rules shall pass all test cases of the abstract test suite in B.1. All encoding rules shall pass all test cases of the abstract test suite in B.2 and/or B.3.

Table 1 — Conformance classes related to encoding rules

Conformance class	Subclause of the abstract test suite
All encoding rules	B.1
Encoding rule with instance conversion	B.2
Encoding rule with schema conversion	B.3

2.3 Conformance classes related to encoding services

All encoding services shall pass all test cases of the abstract test suite in B.4. Depending on the capabilities of the encoding service, it shall pass all test cases of additional conformance classes in accordance with Table 2.

Table 2 — Conformance classes related to encoding services

Conformance class	Subclause of the abstract test suite
All encoding services	B.4
Generic encoding service	B.5
Service that encodes data	B.6
Service that decodes data	B.7
Service that generates an output data structure schema	B.8

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 10646:2011, *Information technology — Universal Coded Character Set (UCS)*

ISO/TS 19103:2005, *Geographic information — Conceptual schema language*

ISO 19109:2005, *Geographic information — Rules for application schema*

Extensible Markup Language (XML) 1.0, W3C Recommendation. Available at <http://www.w3.org/TR/REC-xml>

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1

application schema

conceptual schema (4.5) for **data** (4.8) required by one or more applications

[ISO 19101:2002, 4.2]

NOTE An application schema describes the content, the structure and the constraints applicable to **information** (4.22) in a specific application domain.

4.2

character

member of a set of elements that is used for the representation, organization, or control of **data** (4.8)

[ISO/IEC 2382-1:1993, 01.02.11]

4.3

code

representation of a label according to a specified scheme

4.4**conceptual model**

model (4.27) that defines concepts of a **universe of discourse** (4.33)

[ISO 19101:2002, 4.4]

4.5**conceptual schema**

formal description of a **conceptual model** (4.4)

[ISO 19101:2002, 4.5]

4.6**conceptual schema language**

formal language based on a conceptual formalism for the purpose of representing **conceptual schemas** (4.5)

[ISO 19101:2002, 4.6]

EXAMPLES UML, EXPRESS, IDEF1X.

NOTE A conceptual schema language may be lexical or graphical.

4.7**conversion rule**

rule for converting instances in the input **data** (4.8) structure to instances in the output data structure

4.8**data**

reinterpretable representation of **information** (4.22) in a formalized manner suitable for communication, interpretation, or processing

[ISO/IEC 2382-1:1993, 01.01.02]

4.9**data interchange**

delivery, receipt and interpretation of **data** (4.8)

4.10**data transfer**

movement of **data** (4.8) from one point to another over a **medium** (4.26)

NOTE Transfer of **information** (4.22) implies transfer of data.

4.11**data type**

specification of a **value domain** (4.34) with operations allowed on values in this domain

[ISO/TS 19103:2005, 4.1.5]

EXAMPLES Integer, Real, Boolean, String and Date.

NOTE A data type is identified by a term, e.g. Integer. Values of the data types are of the specified value domain, e.g. all integer numbers between -65537 and 65536. The set of operations can be +, -, * and / and is semantically well defined. A data type can be simple or complex. A simple data type defines a value domain where values are considered atomic in a certain context, e.g. Integer. A complex data type is a collection of data types that are grouped together. A complex data type may represent an object and can, thus, have identity.

4.12**dataset**

identifiable collection of **data** (4.8)

[ISO 19115:2003, 4.2]

4.13**encoding**

conversion of **data** (4.8) into a series of **codes** (4.3)

4.14**encoding rule**

identifiable collection of **conversion rules** (4.7) that define the **encoding** (4.13) for a particular **data** (4.8) structure

EXAMPLES XML, ISO 10303-21, ISO/IEC 8211.

NOTE An encoding rule specifies the types of data being converted as well as the syntax, structure and **codes** (4.3) used in the resulting data structure.

4.15**encoding service**

software component that has an **encoding rule** (4.14) implemented

4.16**feature**

abstraction of real world phenomena

[ISO 19101:2002, 4.11]

NOTE A feature may occur as a type or an instance. Feature type or feature instance is used when only one is meant.

4.17**file**

named set of records stored or processed as a unit

[ISO/IEC 2382-1:1993, 01.08.06]

4.18**geographic data**

data (4.8) with implicit or explicit reference to a location relative to the Earth

[ISO 19109:2005, 4.12]

4.19**geographic information**

information (4.22) concerning phenomena implicitly or explicitly associated with a location relative to the Earth

[ISO 19101:2002, 4.16]

4.20**identifier**

linguistically independent sequence of **characters** (4.2) capable of uniquely and permanently identifying that with which it is associated

[ISO 19135:2005, 4.1.5]

4.21**identification convention**

set of rules for creating **identifiers** (4.20)

4.22**information**

knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning

[ISO/IEC 2382-1:1993, 01.01.01]

4.23**instance model**

representation **model** (4.27) for storing **data** (4.8) according to an **application schema** (4.1)

4.24**interface**

⟨UML⟩ named set of operations that characterize the behaviour of an element

[ISO/IEC 19501]

4.25**interoperability**

capability to communicate, execute programs, or transfer **data** (4.8) among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units

[ISO/IEC 2382-1:1993, 01.01.47]

4.26**medium**

substance or agency for storing or transmitting **data** (4.8)

EXAMPLES Compact disc, internet^[1], radio waves, etc.

4.27**model**

abstraction of some aspects of reality

[ISO 19109:2005, 4.14]

4.28**schema**

formal description of a **model** (4.27)

[ISO 19101:2002, 4.25]

4.29**schema model**

representation **model** (4.27) for storing **schemas** (4.28)

EXAMPLE Representation model for a schema repository.

4.30**stereotype**

⟨UML⟩ new type of modelling element that extends the semantics of the metamodel

[ISO/IEC 19501]

NOTE It is necessary that stereotypes be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure, of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user-defined. Stereotypes are one of three extensibility mechanisms in UML; the others are constraint and tagged value.

4.31

transfer protocol

common set of rules for defining interactions between distributed systems

4.32

transfer unit

collection of **data** (4.8) for the purpose of a **data transfer** (4.10)

NOTE A transfer unit does not have to be identifiable like a **dataset** (4.12).

4.33

universe of discourse

view of the real or hypothetical world that includes everything of interest

[ISO 19101:2002, 4.29]

4.34

value domain

set of accepted values

[ISO/TS 19103:2005, 4.1.15]

EXAMPLE The range 3-28, all integers, any character, enumeration of all accepted values (green, blue, white).

5 Symbols and abbreviated terms

DCE	Distributed computing environment
DUID	Domain unique identifier
HTML	Hypertext markup language
MODIS	Moderate resolution imaging spectroradiometer
POSC	Petroleum Open Standards Consortium
TIFF	Tagged image file format
UCS	Universal multiple-octet coded character set
UML	Unified modelling language
UTF	UCS Transfer format
UUID	Universally unique identifier
XML	Extensible markup language

6 Fundamental concepts and assumptions

6.1 Concepts

The purpose of the set of International Standards known as the “ISO 19100 series” is to enable interoperability between heterogeneous geographic information systems. To achieve interoperability between heterogeneous systems, it is necessary to determine two fundamental issues. The first issue is to define the semantics of the content and the logical structures of geographic data. This shall be done in an application schema. The second issue is to define a system- and platform-independent data structure that can represent data corresponding to the application schema.

The fundamental concepts of data interchange, i.e. the procedure based on the application schema for encoding, delivery, receipt and interpretation of geographic data, are described in 6.2 to 6.6. An overview of the data interchange process is described in 6.2; 6.3 introduces application schemas that allow interpretation of geographic data; 6.4 describes the importance of the encoding rule for producing system-independent data structures; 6.5 describes a software component, called the encoding service, for executing the encoding rule; and 6.6 describes the procedure for delivery and receipt, called the transfer service.

6.2 Data interchange

An overview of a data interchange is shown in Figure 1. System A wants to send a dataset to system B. To ensure a successful interchange, it is necessary that A and B decide on three things: i.e. a common application schema I , which encoding rule R to apply, and what kind of transfer protocol to use. The application schema is the basis of a successful data transfer and defines the possible content and structure of the transferred data, whereas the encoding rule defines the conversion rules for how to code the data into a system-independent data structure.

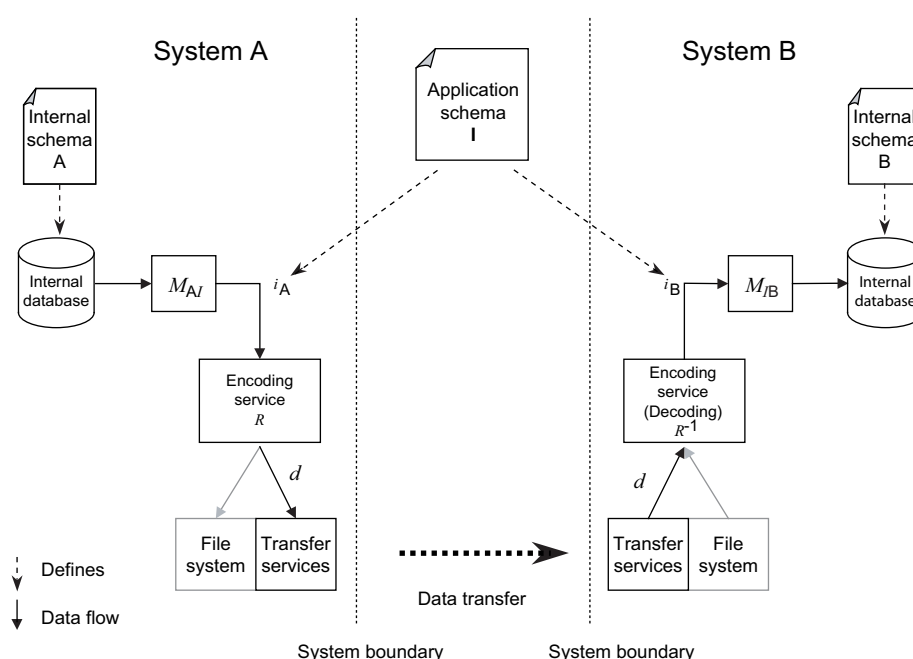


Figure 1 — Overview of data interchange between two systems

Both systems, A and B, store data in an internal database according to an internal schema, but the schemas are usually different, i.e. schema A is not equal to schema B. It is necessary to take the following logical steps in order to transfer a dataset from A's internal database to B's internal database.

- a) The first step for system A is to translate its internal data into a data structure that is in accordance with the common application schema I . Here, this is done by defining a mapping from the concepts of the internal schema to the concepts defined in the application schema and by writing appropriate mapping software to translate the data instances. In Figure 1, this mapping is denoted as M_{AI} . The result is an application-schema-specific data structure, i_A . The data structure is stored in memory or on an intermediate file and is system-dependent and, thus, is not suitable for transfer.
- b) The next step is to use an encoding service, which applies the encoding rule R to create a data structure that is system independent and, therefore, suitable for transfer. This encoded dataset is called d and may be stored in a file system or transferred using a transfer service.
- c) System A then invokes a transfer service to send the encoded dataset d to system B. The transfer service follows a transfer protocol for how to do packaging and how the actual transportation over an on-line or off-line communication medium should take place. It is necessary that both parties agree on the transfer protocol used.
- d) The transfer service on system B receives the transferred dataset, and according to the protocol the dataset is unpacked and stored as an encoded dataset d , e.g. on an intermediate file.
- e) In order to get an application-schema-specific data structure i_B , system B applies the inverse encoding rule R^{-1} to interpret the encoded data.
- f) To use the dataset, it is necessary that B translate the application-schema-specific data structure i_B into its internal database. This is done by defining a mapping from the application schema into its internal schema and by writing software that does the actual translation. In Figure 1 this mapping is denoted M_{IB} .

This International Standard specifies only the requirements for creating encoding rules and the encoding services and not the whole data interchange process. Thus, only steps b) and e) are standardized. Steps a), c), d) and f) use general information technology services.

6.3 Application schema

An application schema is a conceptual schema for applications with similar data requirements. The application schema is the basis of a successful data interchange and defines the possible content and structure of the data. It is also the basis for implementing application-schema-specific data structures for local storage of data.

The application schema used for encoding in compliance with this International Standard shall be written in the UML conceptual schema language, in accordance with ISO/TS 19103 and ISO 19109. These International Standards specify a framework for how to write application schemas. The rules include specifications on how to use standardized schemas to define feature types. It is necessary that both a sender and a receiver of data have access to the application schema.

The application schema shall be accessible to both ends of a data interchange to ensure a successful result. It is necessary that the application schema be transferred before data interchange takes place, so that both the receiver and sender can prepare their systems by implementing mappings and data structures according to the application schema. It may be transferred together with the dataset, or it may be stored in a public place and referenced from the dataset.

The application schema may be interchanged by paper- or electronic-based methods.

6.4 Encoding rule

6.4.1 Concept

An encoding rule is an identifiable collection of conversion rules that defines the encoding for a particular data structure. The encoding rule specifies the data types being converted, as well as the syntax, structure and coding schemes used in the resulting data structure. An encoding rule is applied to application-schema-specific data structures to produce system-independent data structures suitable for transport or storage. In order to define an encoding rule, it is necessary that three important aspects be specified: the input data structure, the output data structure and the conversion rules between the elements of the input and the output data structures. Both the input and output data structures are written using a conceptual schema language and the concepts in the languages are used to define the encoding rule.

6.4.2 Input data structure

The input data structure is an application-schema-specific data structure. The data structure can be thought of as a set of data instances, i.e. $\mathbf{i} = \{i_1, \dots, i_p\}$; see Figure 1. Each data instance, i_k , is an instance of a concept, I_k , defined in an application schema. The application schema defines a set of concepts defined in the application schema $\mathbf{I} = \{I_1, \dots, I_m\}$.

The application schema is a conceptual schema, $\mathbf{c}$, written in a conceptual schema language, $\mathbf{C}$. The conceptual schema defines a set of concepts $\mathbf{c} = \{c_1, \dots, c_m\}$ by instantiating the concepts of the conceptual schema language $\mathbf{C} = \{C_1, \dots, C_p\}$. Since the application schema is a conceptual schema, $\mathbf{c} = \mathbf{I}$.

6.4.3 Output data structure

The output data structure is defined by a schema, $\mathbf{D} = \{D_1, \dots, D_s\}$. D is the schema for the output structure and is not shown in Figure 1. The output data structure can be thought of as a set of data instances, i.e. $\mathbf{d} = \{d_1, \dots, d_q\}$ where each data instance, d_k , is an instance of a concept, D_k .

The schema, $\mathbf{D}$, defines the syntax, structure and coding schemes of the output data structure.

6.4.4 Conversion rules

A conversion rule specifies how a data instance in the input data structure shall be converted to zero, one, or more instances in the output data structure. The conversion rules are defined and based on the concepts of the conceptual schema language, $\mathbf{C}$, and on the concepts of the output data structure schema, $\mathbf{D}$. It is necessary to specify a conversion rule, R_i , for each of the legal combinations of concepts in the conceptual schema language. The set of conversion rules are $\mathbf{R} = \{R_1, \dots, R_n\}$, where R_i is the i -th conversion rule and C_i is the i -th legal combination of instances from the schema language. A conversion table for all possible C_i can be set up, where each C_i maps to a production of instances in the output data structure, $\mathbf{D}$. Figure 2 shows the relationship between the input and output conceptual schema language and the encoding rule.

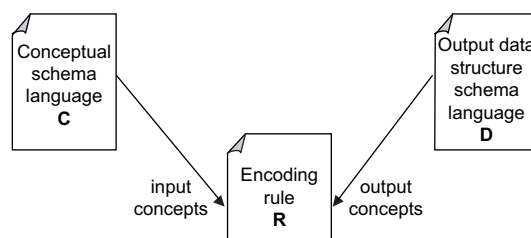


Figure 2 — The encoding rule defines conversion rules from input concepts to output concepts

NOTE The conversion rules are defined based on the two schema languages and not on any particular application schema. This is a generic approach that allows developers to write application-schema-independent encoding services, which can be used for different application schemas as long as the schemas are defined in the same conceptual schema language.

6.5 Encoding service

An encoding service is a software component that has implemented the encoding rule and provides an interface to encoding and decoding functionality. It is an integrated part of data interchange.

Figure 3 presents the details of an encoding service and its relationships to important specification schemas. The encoding service shall be able to read the input data structure and convert the instances to an output data structure and *vice versa*. It shall also be able to read the application schema declarations and write the corresponding output data structure schema. The input data structure is defined by an application schema. The application schema is defined using concepts of the conceptual schema language. The output data structure is also described with a schema, called the data structure schema, which defines the possible content, structure and coding schemes of the output data structure. The data structure schema is described with a schema language. The encoding rule specifies conversion rules at two levels: the first is at the schema level and the second is at the instance level. At the schema level, the conversion rules define a mapping for each of the concepts defined in the application schema to corresponding concepts in the data structure schema. At the instance level, the conversion rules define a mapping for each of the instances in the input data structure to corresponding instances in the output data structure. The instance conversion rules are normally deduced from the schema conversion rules.

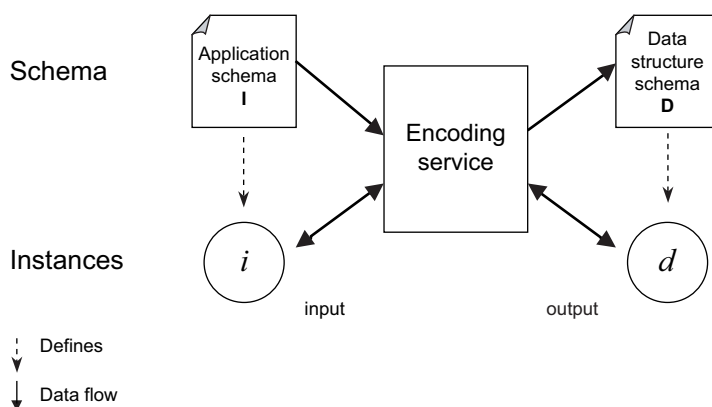


Figure 3 — Overview of the encoding process

An encoding service shall at least provide interfaces for encoding and decoding functionality. Examples of such interfaces are for encoding $d = \text{encode}(i, I)$ and for decoding $i = \text{decode}(d, I)$. Here, i is a reference to an application-schema-specific data structure; I is a reference to the application schema; and d is a reference to the system independent data structure.

6.6 Transfer service

A transfer service is a software component that has implemented one or more transfer protocols that allows data transfer between distributed information systems over off-line or on-line communication media. To successfully transfer data between two systems, it is necessary that the sender and receiver agree on the transfer protocol being used.

Different transfer protocols can be defined. One example is off-line transfer protocols where data are stored on optical or magnetic media and delivered using postal services or other dedicated delivery services. Another example is on-line transfer protocols where data are compressed and included as an email attachment, delivered using a file transfer protocol or transferred using other distributed information technology services which rely on an underlying network service.

This International Standard does not prescribe any preferred transfer protocols.

7 Character repertoire

ISO/IEC 10646 defines an internationally recognized repertoire of characters called the Universal Character Set (UCS) and its character-encoding schemes. The international character set standards defined in ISO/IEC 10646 shall be used in implementing this International Standard.

The character-encoding schemes that can be supported by international profiles of this International Standard are the following:

- a) 8-bit variable size UCS Transfer Format UTF-8;
- b) 16-bit variable size UCS Transfer Format UTF-16;
- c) 16-bit fixed size Universal Character Set UCS-2 (deprecated);
- d) 32-bit fixed size Universal Character Set UCS-4.

International encoding rules that claim conformance with this International Standard shall support one or more of these character-encoding schemes. Within national profiles and system implementations, different character-encoding schemes may be used. The fixed-size character-encoding schemes are often used in database implementations and the variable-size is often used for data interchange purposes.

ISO/IEC 10646 specifies only the repertoire of characters and gives no indication of which language is actually used.

NOTE 1 In cases where it is important to distinguish between different languages in text strings, special mechanisms to indicate the language used can be used.

ISO/IEC 10646 defines mechanisms for creating composite characters. Composite characters are characters produced by superimposing one or more additional characters on a base character. ISO/IEC 10646 defines a set of precomposed characters and their defined decomposition. Since mixing composite characters with their precomposed equivalents can lead to interpretation problems, the use of a composite character if a precomposed character exists is deprecated, i.e. the precomposed character shall always be used.

To summarize, an encoding rule shall

- support one or more character-encoding schemes, and
- not use composite characters if equivalent precomposed characters exist.

EXAMPLE The precomposed character ð has the defined decomposition o".

NOTE 2 For a more detailed description of character normalization, see <http://www.unicode.org/reports/tr15/> and <http://www.w3.org/TR/charmod-norm/>.

NOTE 3 UTF-16, UCS-2 and UCS-4 require information on how to deal with byte ordering, see http://www.unicode.org/faq/utf_bom.html.

8 Generic instance model

8.1 Introduction

A generic instance model is defined in Clause 8. The instance model is a convenient common representation of data when developing encoding services. The instance model is capable of representing data described by application schemas expressed in UML. The instance model represents the application-schema-specific data structure defined in Clause 6 (data structures i_A and i_B in Figure 1). The instance model consists of a dataset (IM_Dataset) that contains a sequence of objects (IM_Object), where an object consists of a sequence of properties (IM_Property). Properties in this context are either attributes or associations; operations are not

included in the general instance model. Each property is encoded according to its data type. The instance model is shown in Figures 4 and 5.

The application schema defines a number of classes and their attributes and associations and it is the basis for generating data representations. A data representation (dataset) contains one or more objects that are structured and encoded according to their class definitions. Clause 8 describes the principles of how to represent objects, their attributes and associations between objects.

The basic unit of information in a dataset is the object. An object shall be an instance of a single concrete class. There are no instances of abstract classes and classes stereotyped as interface. Thus, properties defined by such classes are encoded as part of concrete classes inheriting or realizing them. Each class shall have a unique name within the application schema. The application schema may refer to or use classes defined in standardized schemas or other application schemas. The declaration of these classes shall either be included in the UML model that contains the application schema or accompany the application schema as a separate file.

An object shall contain a set of property values. The object's class defines the properties and they can either be inherited through the "class" supertypes or defined within the class itself. In order to differentiate between the different properties, each property shall have a name that is unique within its class. The property's data type governs the possible values and the multiplicity statement indicates the number of instances of the attribute in an instantiated object.

An object has a corresponding class, defined in an application schema or standardized schema, which defines the possible attributes and associations that are necessary to represent the state of the object. An IM_Object refers to its class by the "class" attribute, it shall be identified within the context of a dataset by its unique identifier "id", and may be universally uniquely identified within a defined universe, application domain or name space, by its "duid" attribute.

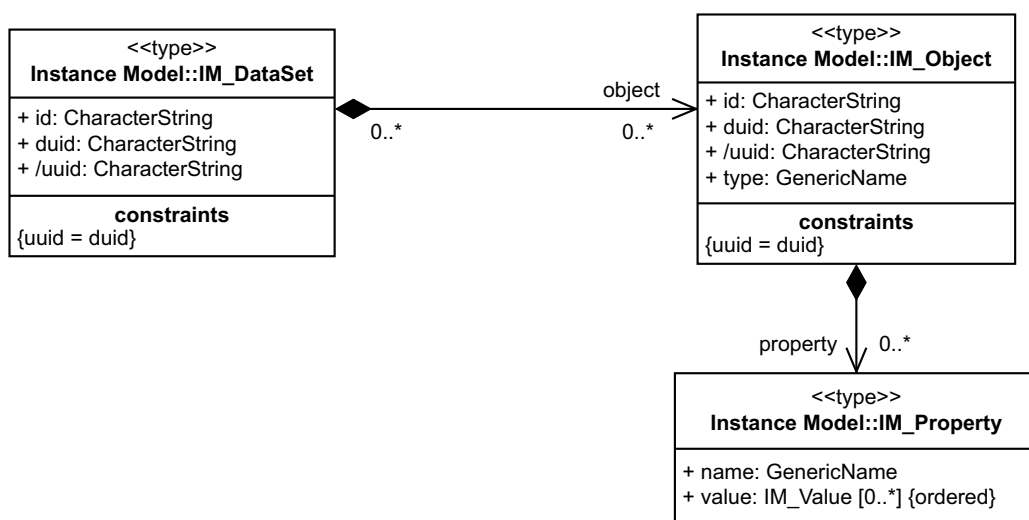


Figure 4 — Instance model — Dataset, object and property

The attributes defined by the class and the association ends navigable from the class are mapped to a set of properties. A property (IM_Property) represents a name with an ordered collection of values. It can represent an attribute or association end. The property name shall correspond to the attribute name or the target role name of an association. A value (IM_Value) represents a property value.

Null values may be given either explicitly or implicitly. An explicit null value shall be indicated by an instance of the corresponding IM_Property with a given nilReason value. An implicit null value is indicated if the corresponding IM_Property instance is missing.

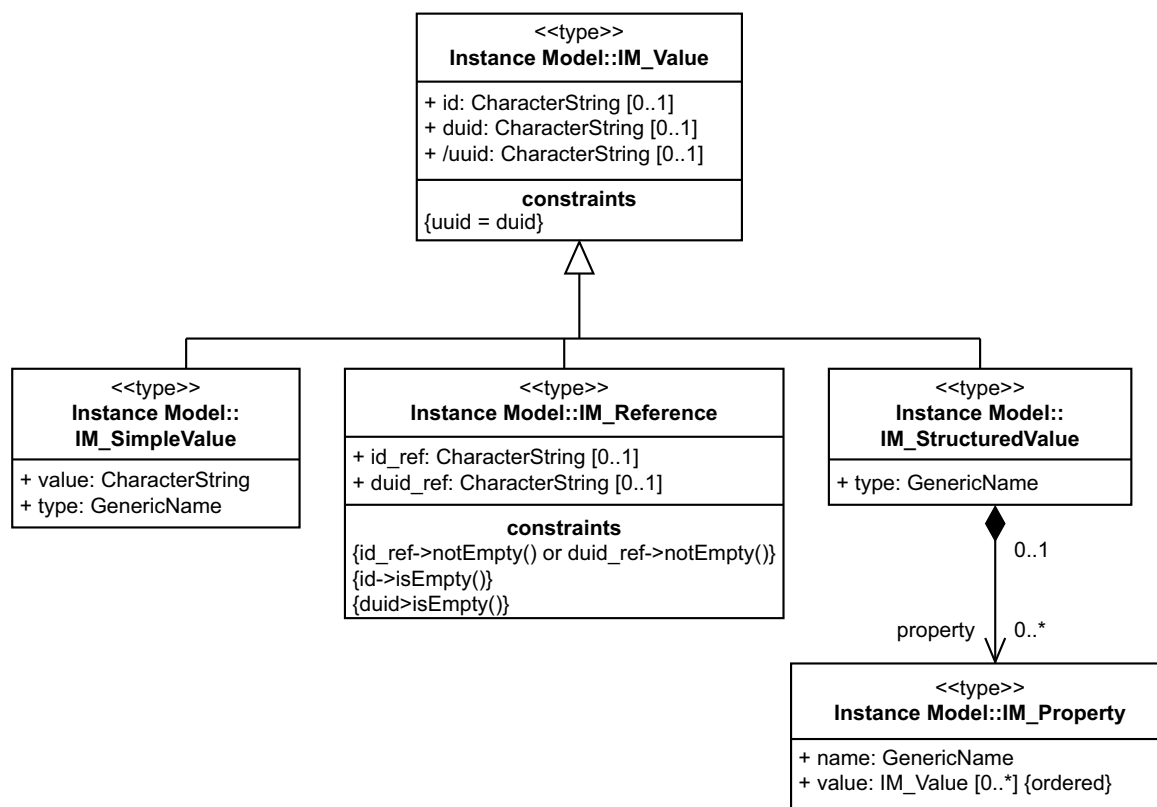


Figure 5 — Instance model — Value types

The three value types are defined as follows.

- IM_SimpleValue represents a value of simple content.

EXAMPLE An integer or a character string.

- IM_Reference represents a link or reference to a target object. The target object may be located in the same transfer unit or another one. A unique identifier (id_ref) targets an object located within the same transfer unit. A domain unique identifier (duid_ref) targets an object located within the context of an application domain.
- IM_StructuredValue represents a data type value with complex content [a sequence of properties (IM_Property)].

An object may, through its associations, be linked (or refer) to one or more objects. UML defines three different types of associations, called association, aggregation and composition, with different semantics; see ISO/TS 19103 for further details. There are, in general, two strategies for representing associations: either encapsulated as a part of the objects or detached from the objects as separate association objects or relational tables.

The encapsulated representation strategy splits an association into a source object property and a target object property. These two link properties then each point to the other object. The link property contains references to its target objects or, in the case of a composition, the target objects themselves. A link property is identified by the role name close to the target class and has a corresponding multiplicity. If the role name is missing or if the association is not navigable from the source object, then there shall be no link property. The two link property values should be consistent in that the referential integrity constraint is enforced. That is, if a source object is referring to another target object through a link property and the target object has a bi-directional association to the source object, the target object shall have a corresponding link property that is referring back to the source object. If a link property is encoded as a reference or an embedded object or not encoded at all, it is defined by a concrete encoding rule (such as Annex A).

8.2 Relation between UML and the instance model

Table 3 gives a summary of the relation between UML and the instance model.

Objects based on classes that have supertypes shall contain all the properties and association ends of their class and of their supertypes. Thus, all attributes and association ends shall be copied from the supertypes and are considered to be a part of the object. Attribute and association end names shall be the way of accessing the values of the attributes and they shall therefore be unique within the class.

Operations and constraints shall not be mapped to the instance model.

Table 3 — Summary of relationship between UML and the instance model

UML concept	Instance model
Package	N/A ^a
Class	
Stereotype	
<<Interface>>	IM_Object
<<DataType>>	IM_SimpleValue or IM_StructuredValue
<<Union>>	IM_StructuredValue
<<Enumeration>>	IM_SimpleValue
<<CodeList>>	IM_SimpleValue or IM_StructuredValue
<<Type>>	IM_Object
<<FeatureType>>	IM_Object
NONE	IM_Object
any other stereotype as defined by the UML profile used	as defined by the encoding rule specification
Attribute	IM_Property with IM_Value according to attribute type; either IM_SimpleValue, IM_Reference or IM_StructuredValue
Association	IM_Property with IM_Reference
Aggregation	IM_Property with IM_Reference
Composition	IM_Property with IM_Value according to target type; either IM_Reference (target type is a class) or IM_SimpleValue or IM_StructuredValue (target type is a data type)
Generalization	The object according to the sub-class carries all the properties that the sub-class inherits from the super-classes.
Operation	N/A
Constraint	N/A
^a N/A stands for not applicable.	

9 Encoding rules

9.1 Introduction

The requirements for specifying encoding rules are defined in 9.2 to 9.6. An encoding rule describes conversion rules for transforming data from an input data structure to an output data structure. Schemas shall be described for both the input and the output data structure. The schema for the input data structure is called an application schema.

An encoding rule shall in general specify the following:

- a) general encoding requirements (9.2):
 - 1) application schemas and schema language,

- 2) order of bits within each byte, and bytes within a word (where applicable),
 - 3) character repertoire and encoding,
 - 4) necessary exchange metadata,
 - 5) dataset and object identification convention;
- b) input data structure (9.3):
- 1) data structure used to pass data according to an application schema (data structures i_A and i_B in Figure 1) to the encoding service, called the instance model,
 - 2) how the instance model is related to the application schema;
- c) output data structure used, called the exchange format (9.4);
- d) conversion rules, called the mapping, for converting data in the instance model to the exchange format (9.5):
- 1) conversion rules for encoding,
 - 2) if necessary, conversion rules for decoding;
- e) sufficient examples of abstract data, application of conversion rules and encoded data (9.6).

9.2 General encoding requirements

9.2.1 Application schema and schema language

The encoding rule shall specify the schema language used to define application schemas and describe how an application schema is organized.

9.2.2 Bit and byte ordering

If the encoding rule specifies a binary encoding, it shall specify the order of the bits within each byte, and the order and number of bytes within any multi-byte structure (word).

NOTE These are general rules that apply to text and binary encoding rules. Even text-based encodings (such as UTF-16, UCS-2, UCS-4) require a specification of byte ordering (see http://www.unicode.org/faq/utf_bom.html).

9.2.3 Character repertoire and encoding

The character repertoire defines the characters that are allowed. The character repertoire doesn't define the language of the data. The same character repertoire can be encoded in different ways.

If more than one language is required, the application schema shall model an attribute so that the actual language of encoded text can be identified.

NOTE See class LocalizedCharacterString in ISO/TS 19103.

The encoding rule shall specify the character repertoire used and the encoding of the characters.

9.2.4 Exchange metadata

Exchange metadata are metadata about the encoded data structure. They may describe the originator of the dataset, refer to metadata information about the dataset, refer to the application schema and provide information about the encoding rule applied.

The encoding rule shall specify the exchange metadata and how it accompanies the encoded data structure.

Figure 6 illustrates an example of an ExchangeMetadata class and a corresponding EncodingRule class. The CI_Citation class is defined in ISO 19115. An exchange file normally contains one instance of the ExchangeMetadata class. The “datasetCitation” attribute contains information about the originator of the dataset, the “metadataCitation” attribute may contain information about where to find metadata about the dataset, and the “applicationSchemaCitation” attribute may hold a reference to the application schema. The EncodingRule’s “encodingRuleCitation” attribute identifies the encoding rule applied and a description of how the rule was applied in this particular case. It may also contain information about the encoding service tool used to encode the dataset, where “toolName” and “toolVersion” indicate the name of the encoding service tool and its version number.

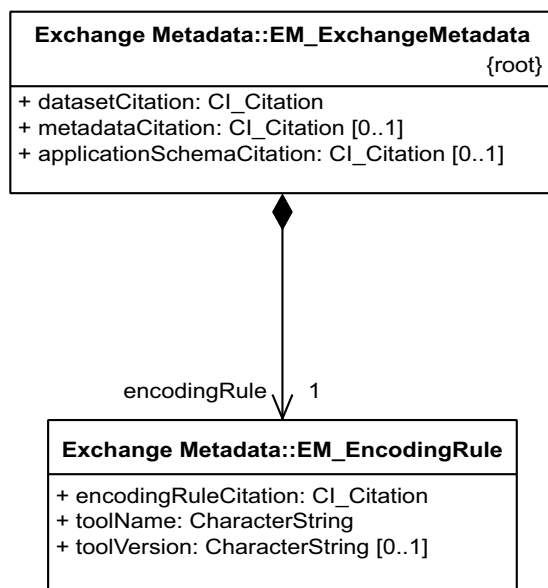


Figure 6 — Example of exchange metadata

9.2.5 Transfer unit

9.2.5.1 Granularity and structure

It is important to define the granularity and the structure of a transfer unit so that it can be encoded and decoded efficiently. An object is here considered as the basic unit of information and it is necessary that the different types of objects be identified. The objects in a transfer unit may be structured sequentially and/or hierarchically. An object may be internally structured as a sequence of attributes, it may contain references to other objects and it may also be composed of other objects in a hierarchical manner.

An object may be split into different fragments, or its properties may be merged with other objects. Whatever changes in structure an encoding rule specifies, the instance conversion shall be unambiguously reversible.

The encoding rule shall specify the following:

- what an object is and the different types of object;
- the structure of an object;
- the structure of a dataset.

9.2.5.2 Object identification

Objects may be assigned identifiers that allow their unique identification within a particular context. Two different contexts should be considered.

- a) The first context is a transfer unit. Here the object identifiers are unique within a particular unit of data transfer. These identifiers allow objects to refer to other objects within the context of that one transfer unit. The identifiers may be assigned to the objects when they are inserted into the transfer unit and are transient in nature.
- b) The second context is an application domain. An application domain defines a universe and an identification convention called domain unique identifiers (DUIDs). A DUID is assigned to an object when it is created and is stable over the entire life span of the object. A DUID of a deleted object shall not be used again. A DUID is mostly used for implementation reasons. DUIDs are required for long term distributed data management and for realizing update mechanisms. These identifiers are also called persistent identifiers. A special name server may be used to resolve persistent identifiers. The identifiers shall be unique within a well-defined, limited universe defined by an application domain.

An application domain may use additional properties to identify the real-world phenomena represented by the object (e.g. a parcel number). These additional properties are not considered object identifiers in the scope of this specification.

An encoding rule may change the layout of the data, for example factoring out common character strings. To support this, types without identity, according to the application schema, may get a fragment identifier. These fragment identifiers are in any case transient (only valid inside the transfer unit), and are never a DUID.

The encoding rules shall specify the following:

- the different object identification mechanisms used;
- their internal structure.

9.3 Input data structure

The input data structure is called the instance model and it is an instrument for reasoning about application data, for defining conversion rules and for expressing examples. The instance model shall be capable of representing data according to the specification in the application schema. It may be specific to a particular application schema or capable of representing data according to any schema. It may be abstract in that it does not have to be implemented in order to realize an encoding service.

The encoding rule shall reference the general instance model as defined in Clause 8 or specify an instance model and its relation to the application schema.

This International Standard does not mandate a specific implementation of the instance model or how data are passed from/to the encoding service.

9.4 Output data structure

The output data structure defines how data is structured and represented in an exchange file. A schema may accompany the output data structure.

The encoding rule shall specify the output data structure and, if present, the output data structure schema.

9.5 Conversion rules

A conversion rule specifies how a data instance in the input data structure is converted to a data instance in the output data structure. Two sets of conversion rules may exist. The first one is the set of schema conversion rules that define a mapping from the UML schema to the schema of the output data structure. The second one is the set of instance conversion rules that define a mapping from instances in the instance model to instances in the resulting data model. Figure 7 shows the different conversion rules.

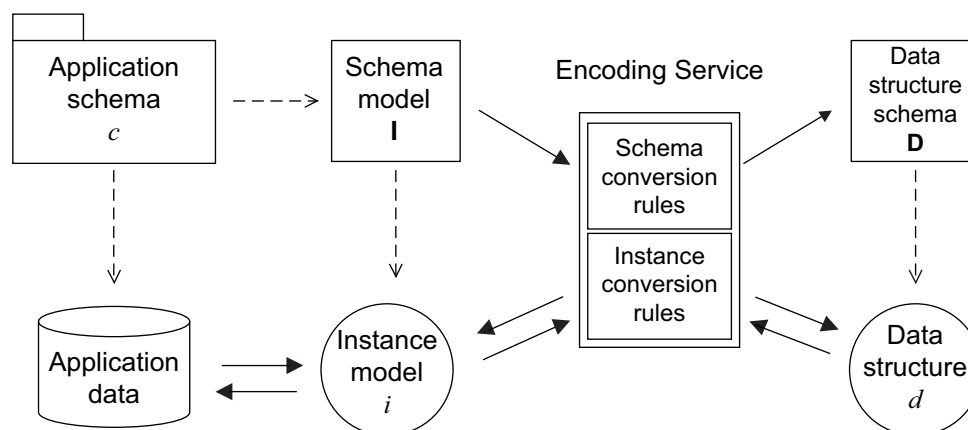


Figure 7 — Conversion rules

The encoding rule shall specify the following:

- schema conversion rules;
- instance conversion rules.

9.6 Examples

Examples are important for understanding the conversion rules and for testing encoding services.

The encoding rule shall provide the following:

- examples that illustrate conversion rules;
- test data that can be used in implementation of encoding services.

10 Encoding service

An encoding service is a software component that implements the encoding rule and provides an interface to its functionality.

An encoding service shall provide an interface to its functionality through one or more interface specifications.

An encoding service shall provide one or more of the following:

- capability to encode data according to the instance conversion rules;
- capability to decode data according to the instance conversion rules;
- capability to create an output data structure schema according to the schema conversion rules.

This International Standard does not mandate a specific interface to the encoding service.

An encoding service may provide only encoding or decoding capabilities.

Figure 8 is an example of an XML-based encoding service.

NOTE Figure 8 shows an encoding service with one interface specification. It supports three operations: generateXMLSchema, encode and decode. The “generateXMLSchema” operation can be used to generate an XML Schema file. This operation takes a schema model as input parameter and produces an XMLStream object as a result. The “encode” operation can be used to generate an XML document. It takes a schema model and an instance model as input parameters and returns an XMLStream object. The “decode” operation can be used to interpret an XML document. It takes a schema model and an XMLStream object as input and returns an instance model.

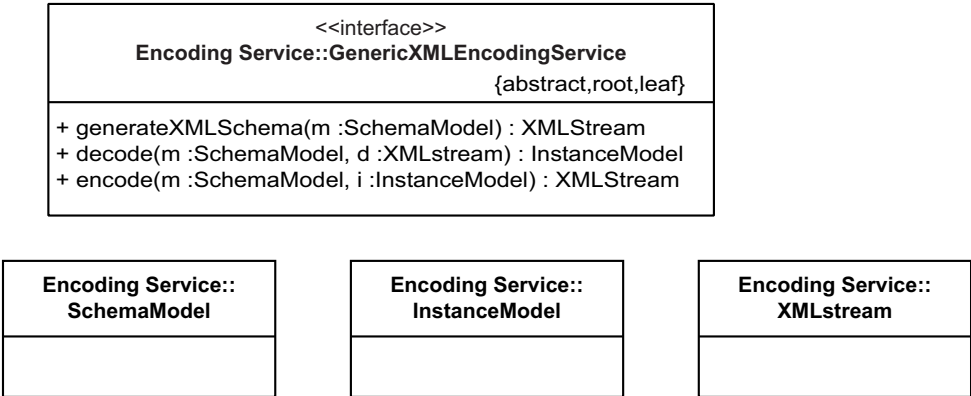


Figure 8 — Example encoding service interface