International Standard

ISO 8000-118

First edition
2025-03

Data quality —

Part 118:
Application of ISO 8000-115 to natural location identifiers

Reference number
ISO 8000-118:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data*.

A list of all parts in the ISO 8000 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

0.1 Foundations of the ISO 8000 series

Digital data deliver value by enhancing all aspects of organizational performance including:

— operational effectiveness and efficiency;

— safety and security;

— reputation with customers and the wider public;

— compliance with statutory regulations;

— innovation;

— consumer costs, revenues and stock prices.

In addition, many organizations are now addressing these considerations with reference to the United Nations Sustainable Development Goals [1].

The influence on performance originates from data being the formalized representation of information [2]. This information enables organizations to make reliable decisions. This decision making can be performed by human beings directly and also by automated data processing including artificial intelligence systems.

Through widespread adoption of digital computing and associated communication technologies, organizations become dependent on digital data. This dependency amplifies the negative consequences of lack of quality in these data. These consequences are the decrease of organizational performance.

The biggest impact of digital data comes from two key factors:

— the data having a structure that reflects the nature of the subject matter;

EXAMPLE 1 A research scientist writes a report using a software application for word processing. This report includes a table that uses a clear, logical layout to show results from an experiment. These results indicate how material properties vary with temperature. The report is read by a designer, who uses the results to create a product that works in a range of different operating temperatures.

— the data being computer processable (machine readable) rather than just being for a person to read and understand.

EXAMPLE 2 A research scientist uses a database system to store the results of experiments on a material. This system controls the format of different values in the data set. The system generates an output file of digital data. This file is processed by a software application for engineering analysis. The application determines the optimum geometry when using the material to make a product.

ISO 9000[2] explains that quality is not an abstract concept of absolute perfection. Quality is actually the conformance of characteristics to requirements. This conformance means that any item of data can be of high quality for one purpose but not for a different purpose. The quality is different because the requirements are different between the two purposes.

EXAMPLE 3 Time data are processed by calendar applications and also by control systems for propulsion units on spacecraft. These data include start times for meetings in a calendar application and activation times in a control system. These start times require less precision than the activation times.

1) https://sdgs.un.org/goals

2) ISO 8000-2[1] defines information as “knowledge concerning objects, such as facts, events, things, processes, or ideas, including concepts, that within a certain context has a particular meaning”.

The nature of digital data is fundamental to establishing requirements that are relevant to the specific decisions that are made by each organization.

EXAMPLE 4 ISO 8000-1[3] identifies that data have syntactic (format), semantic (meaning) and pragmatic (usefulness) characteristics.

To support the delivery of high-quality data, the ISO 8000 series addresses:

— data governance, data quality management and maturity assessment;

EXAMPLE 5 ISO 8000-61[4] specifies a process reference model for data quality management.

— creating and applying requirements for data and information;

EXAMPLE 6 ISO 8000-110[5] specifies how to exchange characteristic data that are master data.

— monitoring and measuring information and data quality;

EXAMPLE 7 ISO 8000-8[6] specifies approaches to measuring information and data quality.

— improving data and, consequently, information quality;

EXAMPLE 8 ISO/TS 8000-81[7] specifies an approach to data profiling, which identifies opportunities to improve data quality.

— issues that are specific to the type of content in a data set.

EXAMPLE 9 ISO/TS 8000-311[8] specifies how to address quality considerations for product shape data.

Data quality management covers all aspects of data processing, including creating, collecting, storing, maintaining, transferring, exploiting and presenting data to deliver information.

Effective data quality management is systemic and systematic, requiring an understanding of the root causes of data quality issues. This understanding is the basis for not just correcting existing nonconformities but for also implementing solutions that prevent future reoccurrence of those nonconformities.

EXAMPLE 10 If a data set includes dates in multiple formats including “yyyy-mm-dd”, “mm-dd-yy” and “dd-mm-yy”, then data cleansing can correct the consistency of the values. Such cleansing requires additional information, however, to resolve ambiguous entries (such as, “04-05-20”, which can be a representation of many different dates including 4 May 2020, 5 April 2020 and 20 May 2004). The cleansing also cannot address any process issues and people issues, including training, that have caused the inconsistency.

0.2 Understanding more about the ISO 8000 series

ISO 8000-1[3] provides a detailed explanation of the structure and scope of the whole ISO 8000 series.

ISO 8000-2[1][3)] specifies the single, common vocabulary for the ISO 8000 series. This vocabulary is ideal reading material by which to understand the overall subject matter of data quality. ISO 8000-2[1] presents the vocabulary structured by a series of topic areas (for example, terms relating to quality and terms relating to data and information).

ISO has identified ISO 8000-1[3], ISO 8000-2[1] and ISO 8000-8[6] as horizontal deliverables[4)].

0.3 Role of this document

As a contribution to the overall capability of the ISO 8000 series, this document specifies how to generate an unambiguous natural location identifier.

Organizations can use this document on its own or in conjunction with other parts of the ISO 8000 series.

3) The content is available on the ISO Online Browsing Platform: https://www.iso.org/obp

4) Deliverable dealing with a subject relevant to a number of committees or sectors or of crucial importance to ensure coherence across standardization deliverables.

This document supports activities that affect:

- one or more information systems;
- data flows within the organization and with external organizations;
- any phase of the data life cycle.

Annex A contains an identifier that conforms to ISO/IEC 8824-1[9]. The identifier unambiguously identifies this document in an open information system.

0.4 Benefits of the ISO 8000 series

By implementing parts of the ISO 8000 series to improve organizational performance, an organization achieves the following benefits:

- objective validation of the foundations for digital transformation of the organization;
- a sustainable basis for data in digital form becoming a fundamental asset class that the organization relies on to deliver value;
- securing evidence-based trust from other parties (including supply chain partners and regulators) about the repeatability and reliability of data and information processing in the organization;
- portability of data with resulting protection against loss of intellectual property and re-usability across the organization and applications;
- effective and efficient interoperability between all parties in a supply chain to achieve traceability of data back to original sources;
- readiness to acquire or supply services where the other party expects to work with common understanding of explicit data requirements.

Data quality —

Part 118:
Application of ISO 8000-115 to natural location identifiers

1 Scope

This document specifies requirements for natural location identifiers. These requirements supplement those of ISO 8000-115[10].

The following are within the scope of this document:

— requirements for the prefix element and a single sub-domain element in a location identifier;

— requirements for representing latitude, longitude and storey or elevation of a location.

The following are outside the scope of this document:

— methods to identify latitude, longitude and elevation.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 8000-2, *Data quality — Part 2: Vocabulary*

ISO 8000-115, *Data quality — Part 115: Master data: Exchange of quality identifiers: Syntactic, semantic and resolution requirements*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 8000-2 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

4 Fundamental principles and assumptions

4.1 Types of identifiers

4.1.1 General

Organizations issue identifiers for many purposes. Identifiers are one of two types:

a) controlled identifiers (see 4.1.2);

b) natural identifiers (see 4.1.3).

4.1.2 Controlled identifiers

Controlled identifiers are issued by an organization that maintains the characteristic data associated with the identifier. These identifiers are, therefore, managed by the organization, which is also responsible for maintaining an identifier registry.

EXAMPLE 1 Controlled identifiers include part numbers, batch numbers and serial numbers, each of which are issued by manufacturers.

EXAMPLE 2 Controlled identifiers include personal identifiers issued by governments, universities and companies.

Each controlled identifier depends on completion of a master data record containing characteristic data to differentiate the identified item within a particular domain. This dependence means resolution of the identifier requires access to the underlying master data that are controlled by the issuing organization.

A controlled identifier is created by, and therefore belongs to, the issuing organization.

4.1.3 Natural identifiers

A natural identifier derives from the characteristics of the identified item, where those characteristics make the item unique within a particular domain. The identifier can be created by applying an algorithm to the characteristics of the item.

EXAMPLE 1 Natural identifiers include life science identifiers, which are persistent, globally unique identifiers for biological objects.

EXAMPLE 2 Natural identifiers include those generated by cryptographic hash functions (to create document and virus signatures) and cyclic redundancy check digits (error-detecting codes to highlight changes to raw data in data transfers).

As a natural identifier derives from accessible information, the identifier is not subject to the control of one owning organization.

4.2 Ownership and usability of identifiers

Controlled identifiers belong to the issuing organization.

Unless the controlled identifier is in the public domain, data containing controlled identifiers can end up becoming a joint work under copyright law.

The issuing organization for a controlled identifier can, for instance, restrict use of the identifier only to retrieve licensed content provided by the issuer of the identifier.

For natural identifiers, if the algorithm used to generate it is proprietary, then the resulting identifier is proprietary. In this case, both the use of the algorithm and the resulting identifier will be subject to license.

If the natural identifier uses an algorithm that is in the public domain, published as an open standard or can be used without license, then the resulting identifier is a public or open identifier that is usable without license.

4.3 Geographic location identifiers

A geographic location identifier represents the identity of a geographical point location.

In order to identify a geographic point location on Earth, a unique, natural identifier can be generated from the corresponding coordinate of the location within a geodetic coordinate reference system.

Geographic location identifiers are bound to a particular geodetic coordinate reference system as a datum (which is also known as a “reference frame”).

NOTE There are multiple realizations of geodetic datums. These realizations include the series of the World Geodetic System (WGS)[11], which is used by the Global Positioning System (GPS)[12], the International Terrestrial Reference Frame (ITRF), which is a precise scientific realization[13]; and the European Terrestrial Reference Frame (ETRF), which is the EU-recommended reference frame[14].

Clients of global navigation satellite systems have become pervasive, not only in mapping devices and surveying equipment but also in mobile phones and Internet-of-Things devices. These clients can easily calculate geographic location identifiers on demand by using a simple, open algorithm.

5 Natural location identifier representation

5.1 General

The natural location identifier is a geographic location identifier that provides a unique identifier for a geographic point location in three dimensions.

The natural location identifier is a natural identifier because the identifier is generated from intrinsic characteristics of a geographic point location. These characteristics are the coordinates of the location within a geodetic coordinate system. The characteristics are not assigned by an individual person or an organization.

In practice, the natural location identifier provides a lossless conversion of the coordinates for a geographic point location, allowing the translation of the identifier to the original coordinates within the chosen geodetic coordinate system and corresponding reference frame.

5.2 Representation

This document uses the following characteristics as the basis for the natural location identifier:

— the coordinates (latitude and longitude) of the location according to the World Geodetic System 1984[11];

— elevation coordinates, in the form of either ground-level distance or a storey identifier.

A natural location identifier consists of the following elements:

— a prefix element as the string “ISO”;

— a sub-domain element as the string “.NLI”;

— an identifier element consisting of the character “:” followed by the encoding of the coordinates and elevation using the algorithms specified by 5.3 and Clause 6.

NOTE 1 The identifier conforms to requirements of ISO 8000-115[10] by consisting of a prefix element, an optional sub-domain element and an identifier element.

For an example of the encoding and representation of a natural location identifier described in this document, see Annex C.

NOTE 2 Data tables used for encoding values of the natural location identifier provided in Annex B are provided in the public domain and not subject to copyright (see Annex E).

5.3 Algorithms

5.3.1 General

A core set of algorithms enable generation of a natural location identifier (see Table 5-1).

Table 5-1 — Necessary algorithms to generate natural location identifiers

Algorithm	Description
Identifier(latitude, longitude, elevation, elevationType)	Generates the natural location identifier from a geographic point location representation (see 5.3.2)
EncodePoint(latitude, longitude)	Encodes a geographic point location of a geodetic coordinate reference system into a string of characters (see 5.3.3)
EncodeLatitude(latitude)	Encodes the latitude of a geographic point location of a geodetic coordinate reference system into a string of characters (see 5.3.4)
EncodeLongitude(longitude)	Encodes the longitude of a geographic point location of a geodetic coordinate reference system into a string of characters (see 5.3.5)
EncodeElevation(elevation, elevationType)	Encodes an elevation coordinate of a discrete elevation reference system into a string of characters (see 5.3.6)
EncodeStorey(elevation)	Encodes the storey identifier into a string of characters (see 5.3.7)
EncodeGroundLevel(elevation)	Encodes the ground-level distance into a string of characters (see 5.3.8)
EncodeDecimal(decimalPortion)	Encodes the decimal portion of a decimal number into a string of characters (see 5.3.9)
NOTE See Annex D for a list and a diagram showing the dependencies for all the algorithms specified by this document.	

5.3.2 Identifier(latitude, longitude, elevation, elevationType)

This algorithm generates the natural location identifier from a geographic point location representation (see Table 5-2). The natural location identifier is output as a string.

Table 5-2 — Inputs to the algorithm Identifier

Input	Description
latitude	Number from −90 to +90, in decimal form, representing an angular degree, with a maximum of six decimal places
longitude	Number from −180 to +180, in decimal form, representing an angular degree, with a maximum of six decimal places
elevation	Storey identifier or ground-level distance
elevationType	The elevation type as either the string “storey” or “ground”

The method for this algorithm consists of the following steps:

a) add the string “ISO.NLI:” at the start of the identifier;

b) encode the geographic point location using the EncodePoint algorithm;

c) encode the elevation using the EncodeElevation algorithm;

d) concatenate the string “ISO.NLI:” with the encoded strings for the geographic point location and the elevation.

The pseudocode for the algorithm Identifier is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
Identifier(latitude, longitude, elevation, elevationType)
  return CONCAT(
    "ISO.NLI:",
    EncodePoint(latitude, longitude),
```

```
    EncodeElevation(elevation, elevationType)
  )
```

5.3.3 EncodePoint(latitude, longitude)

This algorithm encodes a geographic point location according to a geodetic coordinate reference system into a string of characters (see Table 5-3). The string is the output of the algorithm. This algorithm is called by the algorithm Identifier (see 5.3.2) for the purpose of encoding a natural location identifier.

Table 5-3 — Inputs to the algorithm EncodePoint

Input	Description
latitude	Number from −90 to +90, in decimal form, representing an angular degree, with a maximum of six decimal places
longitude	Number from −180 to +180, in decimal form, representing an angular degree, with a maximum of six decimal places

The method for this algorithm consists of the following steps:

a) encode latitude using the EncodeLatitude algorithm;

b) encode longitude using the EncodeLongitude algorithm;

c) concatenate the encoded strings of latitude and longitude.

The pseudocode for the EncodePoint algorithm is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodePoint(latitude, longitude)return
CONCAT(EncodeLatitude(latitude),EncodeLongitude(longitude))
```

5.3.4 EncodeLatitude(latitude)

This algorithm encodes the latitude of a geographic point location according to a geodetic coordinate reference system into a string of characters (see Table 5-4). The string is the output of the algorithm. This algorithm is called by the algorithm EncodePoint (see 5.3.3) for the purpose of encoding a geographic point location.

Table 5-4 — Inputs to the algorithm EncodeLatitude

Input	Description
latitude	Number from −90 to +90, in decimal form, representing an angular degree, with a maximum of six decimal places

The method for this algorithm consists of the following steps:

a) normalize the input value by adding 90 to set the midpoint to represent 0 degrees;

b) encode the numeral portion by using the EncodeBase14 algorithm on the numeral portion;

c) encode the decimal portion by using the EncodeDecimal algorithm on the decimal portion;

d) concatenate the encodings of the numeral and decimal portions.

EXAMPLE 1 The latitude encoding for 0 degrees is the string "66".

EXAMPLE 2 The latitude encoding for −90 is the string "00"; for +90, the string is "CC".

EXAMPLE 3 The latitude encoding for −79,25 is padded into the string "−079.250000".

The pseudocode for the algorithm EncodeLatitude is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeLatitude(latitude)if not -90 <= latitude <= 90raise out of bounds errorendifreturn
CONCAT(EncodeLatitudeNumeral(latitude),EncodeDecimal(getDecimal(latitude)))
```

The pseudocode for the algorithm EncodeLatitudeNumeral is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeLatitudeNumeral(latitude)latitudeNumeral = floor(latitude) + 90return
EncodeBase14(latitudeNumeral)
```

NOTE See Clause B.2 for the values returned by this algorithm.

5.3.5 EncodeLongitude(longitude)

This algorithm encodes the latitude of a geographic point location according to a geodetic coordinate reference system into a string of characters (see Table 5-5). The string is the output of the algorithm. This algorithm is called by the algorithm EncodePoint (see 5.3.3) for the purpose of encoding a geographic point location.

Table 5-5 — Inputs to the algorithm EncodeLongitude

Input	Description
longitude	Number from –180 to 180, representing an angular degree, with a maximum of six decimal places

The method for this algorithm consists of the following steps:

a) normalize the value by adding 180 to set the midpoint to represent 0 degrees;

b) encode the numeral portion by using the EncodeBase14 algorithm on the numeral portion;

c) encode the decimal portion by using the EncodeDecimal algorithm on the decimal portion;

d) concatenate the encodings of the numeral and decimal portions.

EXAMPLE 1 The latitude encoding for 0 degrees is the string “99”.

EXAMPLE 2 The latitude encoding for –180 is the string “00”; for +180, the string is “YY”.

The pseudocode for the algorithm EncodeLongitude is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeLongitude(longitude)if not -180 <= latitude <= 180raise out of bounds errorendifreturn
CONCAT(encodeLongitudeNumeral(longitude),EncodeDecimal(getDecimal(longitude)))
```

The pseudocode for the algorithm EncodeLongitudeNumeral is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeLongitudeNumeral(longitude)longitudeNumeral = floor(longitude) + 180return
EncodeBase19(longitudeNumeral)
```

NOTE See Clause B.3 for the values returned by this algorithm.

5.3.6 EncodeElevation(elevation, elevationType)

This algorithm encodes an elevation coordinate according to a discrete elevation reference system into a string of characters (see Table 5-5). The string is the output of the algorithm. This algorithm is called by the algorithm Identifier (see 5.3.2) for the purpose of encoding a natural location identifier.

Table 5-6 — Inputs to the algorithm EncodeElevation

Input	Description
elevation	Storey identifier or ground-level distance
elevationType	Indicates the elevation type as either the string “storey” or “ground”

The method for this algorithm consists of the following steps:

a) if the elevation type is "storey", then the elevation value is a storey identifier and returns the results of the EncodeStorey algorithm;

b) if the elevation type is "ground", then the elevation value indicates ground-level distance and returns the results of the EncodeGroundLevel algorithm.

The pseudocode for the algorithm EncodeElevation is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeElevation(elevation, elevationType)if elevationType = "storey" thenreturn
EncodeStorey(elevation)elsif elevationType = "ground" thenreturn EncodeGroundLevel(elevation)
elseraise unknown elevation type errorendif
```

NOTE While decoding an natural location identifier, if the number of characters is 14, then the elevation portion represents storey information. If the number of characters is 15, then the elevation portion represents ground-level distance information.

5.3.7 EncodeStorey(storey)

This algorithm encodes the storey identifier into a string of characters (see Table 5-7). The string is the output of the algorithm. This algorithm is called by the algorithm EncodeElevation (see 5.3.6) for the purpose of encoding an elevation that is represented by a storey identifier.

Table 5-7 — Inputs to the algorithm EncodeStorey

Input	Description
storey	Identifier of the storey as an integer between −578 and +577

The storey identifier:

— has either a plus or minus sign;

— has a value range of −578 to +577;

— represents a ground floor as "0";

— represents floors above the ground floor using positive numbers;

— represents floors below the ground floor using negative numbers.

The method for this algorithm consists of the following steps:

a) normalize the input by adding 578 to set the midpoint as representing the ground storey;

b) encode the storey identifier using the EncodeBase34 algorithm;

c) return the encoded storey identifier.

EXAMPLE 1 The storey encoding for the ground floor is the string "H0".

EXAMPLE 2 The storey encoding for the fifth floor is the string "H5".

EXAMPLE 3 The storey encoding for the second storey below the ground floor is the string "GW".

The pseudocode for the algorithm EncodeStorey is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeStorey(storey)if not -578 =< storey <= 577raise out of bounds errorendifstorey = storey
+ 578return EncodeBase34(storey)
```

5.3.8 EncodeGroundLevel(elevation)

This algorithm encodes the ground-level distance into a string of characters (see Table 5-8). The string is the output of the algorithm. This algorithm is called by the algorithm EncodeElevation (see 5.3.6) for the purpose of encoding an elevation that is represented by a ground-level distance.

Table 5-8 — Inputs to the algorithm EncodeGroundLevel

Input	Description
groundLevel	Ground-level distance as an integer between –1 000 and +1 000

The ground-level distance is the rounded distance in metres from ground level, where:

— distance above ground level is represented by positive numbers;

— distance below ground level is represented by negative numbers;

— ground-level distance is rounded to the nearest metre;

— values beyond –19 651 and +19 651 are rejected (representing 19 651 m below or 19 651 m above ground level).

Ground level is encoded in base-34 as the midpoint (H00), where the distance is 0 m.

The method for this algorithm consists of the following steps:

a) normalize the input by adding 19 652 , which sets the midpoint to represent the ground level;

b) encode the ground-level distance using the EncodeBase34 algorithm (see 6.2.7);

c) return the encoded storey identifier.

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeGroundLevel(groundLevel)if not -19652 <= groundLevel <= 19651raise out of bounds
errorendifgroundLevel = groundLevel + 19652return EncodeBase34(groundLevel)
```

NOTE See Clause B.5 for values of this encoding.

5.3.9 EncodeDecimal(decimalPortion)

This algorithm encodes the decimal portion of a decimal number into a string (see Table 5-9). The string is the output of the algorithm. This algorithm is called by the algorithms EncodeLatitude (see 5.3.4) and EncodeLongitude (see 5.3.5) for the purpose of encoding the decimal portions of latitude and longitude, respectively.

Table 5-9 — Inputs to the algorithm EncodeDecimal

Input	Description
decimalPortion	Decimal portion of a decimal number

The method for this algorithm consists of the following steps:

a) split the decimal portion into groups of three digits;

b) for each digit group, treat the group of digits as a three-digit integer and encode the number using the EncodeBase32 algorithm, padding the group with one or two "0" characters as suffix to give three digits if the group does not have sufficient digits;

c) concatenate the encoded strings of the digit groups.

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeDecimal(decimalPortion)encoding = ""do forall digitGroupNumber : ∈ every 3 digits in
decimalPortionencoding += EncodeBase32(digitGroupNumber)enddoreturn encoding
```

6 Supporting algorithms

6.1 Common algorithms

The base algorithms are:

— “concat(string1, … stringN)”, which combines the input strings into a single string;

— “floor(number)”, which outputs the largest integer less than or equal to the input number;

— “getDecimal(number)”, which outputs (as a number) the decimal portion of the input number.

NOTE The mechanisms of common algorithms are platform-specific and provide trivial functionality. This document, therefore, does not explicitly specify these algorithms.

6.2 Numeric encoding algorithms

6.2.1 General

The natural location identifier uses numeric encoding to compress the values of the components.

This compression is performed by the following algorithms:

— “EncodeBase14(number)”, which returns a base−14 encoded string of the input number (see 6.2.3);

— “EncodeBase19(number)”, which returns a base−19 encoded string of the input number (see 6.2.4);

— “EncodeBase32(number)”, which returns a base−32 encoded string of the input number (see 6.2.5);

— “EncodeStoreyBase34(number)”, which returns a base−34 encoded string of the input number (see 6.2.6);

— “EncodeGroundBase34(number)”, which returns a base−34 encoded string of the input number (see 5.3.8);

— “EncodeBaseGeneral(number, base, digits, table)”, which returns a base-encoded string of the input number, using the symbols provided by the table (see 5.3.9).

NOTE The encoding algorithms specified by this document are different from and, therefore, not compatible with those specified by RFC 4648[16].

6.2.2 Elimination of visually ambiguous encoding characters

In typical base-n encodings intended for machine-readability, such as the base-32 and base-64 encoding algorithms in RFC 4648[16], the character range of numeric characters “0” to “9” is used together with the Latin alphabet of “A” to “Z” as encoding characters.

However, in an encoding intended to be visually identified by humans, visually ambiguous encoding characters can cause misidentification.

In particular, the following character pairs are visually ambiguous:

— the uppercase letter “I” and the numeric character “1”;

— the uppercase letter “O” and the numeric character “0”.

To prevent confusion on visual inspection, this document specifies encodings that replace all occurrences of:

— the character “I” with the character “Y”;

— the character “O” with the character “Z”.

6.2.3 EncodeBase14(number)

This algorithm generates a base-14 encoded string of the input number (which is an integer between 0 and 180, see Table 6-1). The string consists of two characters. This algorithm is called by the algorithm EncodeLatitudeNumeral (see 5.3.4) for the purpose of encoding the latitude numeral.

Table 6-1 — Inputs to the algorithm EncodeBase14

Input	Description
number	Whole number (i.e. an integer that is not negative) between 0 and 180

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeBase14(number)return EncodeBaseGeneral(number, 14, 2, base14Table)
```

In this pseudocode, the base14Table value is the content of Table 6-2.

Table 6-2 — Value encoding for the algorithm EncodeBase14

Numeric value for encoding (i.e. in decimal)	Encoded value
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D

6.2.4 EncodeBase19(number)

This algorithm generates a base-19 encoded string of the input number (which is an integer between 0 and 360, see Table 6-3). The string consists of two characters. This algorithm is called by the algorithm EncodeLongitudeNumeral (see 5.3.5) for the purpose of encoding the longitude numeral.

Table 6-3 — Inputs to the algorithm EncodeBase19

Input	Description
number	Whole number (i.e. an integer that is not negative) between 0 and 360

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeBase19(number)return EncodeBaseGeneral(number, 19, 2, base19Table)
```

In this pseudocode, the base19Table value is the content of Table 6-4.

Table 6-4 — Value encoding for the algorithm EncodeBase19

Numeric value for encoding (i.e. in decimal)	Encoded value
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	Y
NOTE The value 18 is represented by “Y” rather than the uppercase version of the letter “i” (i.e. “I”) to avoid confusion with the digit representing the number one (i.e. “1”).	

6.2.5 EncodeBase32(number)

This algorithm generates a base-32 encoded string of the input number (which is an integer between 0 and 1 000, see Table 6-5). The string consists of two characters. This algorithm is called by the algorithm EncodeDecimal (see 5.3.9) for the purpose of encoding the decimal portion of a decimal number.

Table 6-5 — Inputs to the algorithm EncodeBase32

Input	Description
number	Whole number (i.e. an integer that is not negative) between 0 and 1 000

The pseudocode for the algorithm is as follows (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeBase32(number)return EncodeBaseGeneral(number, 34, 2, base32Table)
```

NOTE See Clause B.4 for values returned by this algorithm.

In this pseudocode, the base32Table value is the content of Table 6-6.

Table 6-6 — Value encoding for the algorithm EncodeBase32

Numeric value for encoding (i.e. in decimal)	Encoded value
0	0
1	1
NOTE 1 The value 18 is represented by “Y” rather than the uppercase version of the letter “i” (i.e. “I”) to avoid confusion with the digit representing the number one (i.e. “1”). NOTE 2 The value 24 is represented by “Z” rather than the uppercase version of the letter “o” (i.e. “O”) to avoid confusion with the digit representing the number zero (i.e. “0”).	

Table 6-6 *(continued)*

Numeric value for encoding (i.e. in decimal)	Encoded value
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	Y
19	J
20	K
21	L
22	M
23	N
24	Z
25	P
26	Q
27	R
28	S
29	T
30	U
31	V

NOTE 1 The value 18 is represented by “Y” rather than the uppercase version of the letter “i” (i.e. “I”) to avoid confusion with the digit representing the number one (i.e. “1”).

NOTE 2 The value 24 is represented by “Z” rather than the uppercase version of the letter “o” (i.e. “O”) to avoid confusion with the digit representing the number zero (i.e. “0”).

6.2.6 EncodeStoreyBase34(number)

This algorithm generates a base-34 encoded string of the input number (which is an integer between 0 and 1 156, see Table 6-7). The string consists of two characters. This algorithm is called by the algorithm EncodeStorey (see 5.3.7) for the purpose of encoding a storey identifier.

Table 6-7 — Inputs to the algorithm EncodeStoreyBase34

Input	Description
number	Whole number (i.e. an integer that is not negative) between 0 and 1 156

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeStoreyBase34(number)return EncodeBaseGeneral(number, 34, 2, base34Table)
```

NOTE See Clause B.5 for values returned by this algorithm.

In this pseudocode, the base34Table value is the content of Table 6-8.

Table 6-8 — Value encoding for the algorithms EncodeStoreyBase34 and EncodeGroundBase34

Numeric value for encoding (i.e. in decimal)	Encoded value
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	G
17	H
18	Y
19	J
20	K
21	L
22	M
23	N
24	Z
25	P
26	Q
27	R
28	S
29	T
30	U
31	V
32	W
33	X

NOTE 1 The value 18 is represented by “Y” rather than the uppercase version of the letter “i” (i.e. “I”) to avoid confusion with the digit representing the number one (i.e. “1”).

NOTE 2 The value 24 is represented by “Z” rather than the uppercase version of the letter “o” (i.e. “O”) to avoid confusion with the digit representing the number zero (i.e. “0”).

6.2.7 EncodeGroundBase34(number)

This algorithm generates a base-34 encoded string of the input number (which is an integer between 0 and 39 304, see Table 6-9). The string consists of three characters. This algorithm is called by the algorithm EncodeGroundLevel (see 5.3.8) for the purpose of encoding a ground-level distance.

NOTE This algorithm differs from EncodeStoreyBase34 (see 6.2.6) by accepting a larger maximum value and by producing a string of three characters in length.

Table 6-9 — Inputs to the algorithm EncodeGroundBase34

Input	Description
number	Whole number (i.e. an integer that is not negative) between 0 and 39 304

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeGroundBase34(number)return EncodeBaseGeneral(number, 34, 3, base34Table)
```

In this pseudocode, the base34Table value is the content of Table 6-8.

6.2.8 EncodeBaseGeneral(number, base, digits, table)

This algorithm generates output in the form of an encoded string to represent the number input as per the value of the base input and the contents of the table input (see Table 6-10). The string has a length that depends on the value of the digits input.

Table 6-10 — Inputs to the algorithm EncodeBaseGeneral

Input	Description
number	Whole number (i.e. an integer that is not negative)
base	The base in which the output is to represent the input number
digits	Whole number of digits to include in the output
table	Table that specifies the symbols by which to encode the input number into the output

The pseudocode for the algorithm is (where symbols have the meaning specified by ECMA-262[15]):

```
EncodeBaseGeneral(number, base, digits, table)remainderArr = []quotient = numberindex
= 0encoding = ""do forall i: 0, i < digitsremainderArr[i] = 0enddodo while quotient !=
0remainder = quotient % baseremainderArr[index] = remainderquotient = floor(quotient / base)
index = index + 1enddodo forall i : 0, i < digitsencoding += lookup(table, remainderArr[digits
- i - 1])enddoreturn encoding
```

7 Requirements for natural location identifiers

Requirements for natural location identifiers are as follows:

a) The natural location identifier shall conform to ISO 8000-115.

b) The natural location identifier shall have a prefix element of "ISO".

c) The natural location identifier shall have a sub-domain element of ".NLI".

d) The natural location identifier shall have an identifier element that starts with ":", which is followed by the encoding of the coordinates of the location according to the World Geodetic System 1984[11] and of the elevation of the location using the algorithms specified by 5.3 and Clause 6.

8 Conformance

A natural location identifier conforms to this document when Clause 7 is met.

Annex A
(informative)

Document identification

To provide for unambiguous identification of an information object in an open system, the following object identifier is assigned to this document. The meaning of this value is defined in ISO 10303-1[17].

```
{ iso standard 8000 part(118) version(1) }
```

Annex B
(informative)

Encoding tables

B.1 General

The encoding tables in this annex are illustrative, supporting the implementation and testing of the algorithms specified by this document (see 5.3 and 6.2). These algorithms enable the encoding of natural location identifiers and include:

— the EncodeLatitudeNumeral algorithm (see 5.3.4 and Clause B.2);

— the EncodeLongitudeNumeral algorithm (see 5.3.5 and Clause B.3);

— the EncodeDecimal algorithm (see 5.3.9 and Clause B.4);

— the EncodeStoreyBase34 algorithm (see 6.2.6 and Clause B.5).

B.2 Encoding returned by the EncodeLatitudeNumeral algorithm

Table B.2-1 — EncodeLatitudeNumeral encoding in base-14

Second character of the encoded value	First character of the encoded value												
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**
0	−90	−76	−62	−48	−34	−20	−6	+8	+22	+36	+50	+64	+78
1	−89	−75	−61	−47	−33	−19	−5	+9	+23	+37	+51	+65	+79
2	−88	−74	−60	−46	−32	−18	−4	+10	+24	+38	+52	+66	+80
3	−87	−73	−59	−45	−31	−17	−3	+11	+25	+39	+53	+67	+81
4	−86	−72	−58	−44	−30	−16	−2	+12	+26	+40	+54	+68	+82
5	−85	−71	−57	−43	−29	−15	−1	+13	+27	+41	+55	+69	+83
6	−84	−70	−56	−42	−28	−14	0	+14	+28	+42	+56	+70	+84
7	−83	−69	−55	−41	−27	−13	+1	+15	+29	+43	+57	+71	+85
8	−82	−68	−54	−40	−26	−12	+2	+16	+30	+44	+58	+72	+86
9	−81	−67	−53	−39	−25	−11	+3	+17	+31	+45	+59	+73	+87
A	−80	−66	−52	−38	−24	−10	+4	+18	+32	+46	+60	+74	+88
B	−79	−65	−51	−37	−23	−9	+5	+19	+33	+47	+61	+75	+89
C	−78	−64	−50	−36	−22	−8	+6	+20	+34	+48	+62	+76	+90
D	−77	−63	−49	−35	−21	−7	+7	+21	+35	+49	+63	+77	—

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value −77 is encoded as the string "0D".

B.3 Encoding returned by the EncodeLongitudeNumeral algorithm

Table B.3-1 — EncodeLongitudeNumeral encoding in base-19 (–180 to +9)

Second character of the encoded value	First character of the encoded value									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	–180	–161	–142	–123	–104	–85	–66	–47	–28	–9
1	–179	–160	–141	–122	–103	–84	–65	–46	–27	–8
2	–178	–159	–140	–121	–102	–83	–64	–45	–26	–7
3	–177	–158	–139	–120	–101	–82	–63	–44	–25	–6
4	–176	–157	–138	–119	–100	–81	–62	–43	–24	–5
5	–175	–156	–137	–118	–99	–80	–61	–42	–23	–4
6	–174	–155	–136	–117	–98	–79	–60	–41	–22	–3
7	–173	–154	–135	–116	–97	–78	–59	–40	–21	–2
8	–172	–153	–134	–115	–96	–77	–58	–39	–20	–1
9	–171	–152	–133	–114	–95	–76	–57	–38	–19	0
A	–170	–151	–132	–113	–94	–75	–56	–37	–18	+1
B	–169	–150	–131	–112	–93	–74	–55	–36	–17	+2
C	–168	–149	–130	–111	–92	–73	–54	–35	–16	+3
D	–167	–148	–129	–110	–91	–72	–53	–34	–15	+4
E	–166	–147	–128	–109	–90	–71	–52	–33	–14	+5
F	–165	–146	–127	–108	–89	–70	–51	–32	–13	+6
G	–164	–145	–126	–107	–88	–69	–50	–31	–12	+7
H	–163	–144	–125	–106	–87	–68	–49	–30	–11	+8
Y	–162	–143	–124	–105	–86	–67	–48	–29	–10	+9

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value –162 is encoded as the string “0Y”.

Table B.3-2 — EncodeLongitudeNumeral encoding in base-19 (+10 to +180)

Second character of the encoded value	First character of the encoded value								
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**Y**
0	+10	+29	+48	+67	+86	+105	+124	+143	+162
1	+11	+30	+49	+68	+87	+106	+125	+144	+163
2	+12	+31	+50	+69	+88	+107	+126	+145	+164
3	+13	+32	+51	+70	+89	+108	+127	+146	+165
4	+14	+33	+52	+71	+90	+109	+128	+147	+166
5	+15	+34	+53	+72	+91	+110	+129	+148	+167
6	+16	+35	+54	+73	+92	+111	+130	+149	+168
7	+17	+36	+55	+74	+93	+112	+131	+150	+169
8	+18	+37	+56	+75	+94	+113	+132	+151	+170
9	+19	+38	+57	+76	+95	+114	+133	+152	+171
A	+20	+39	+58	+77	+96	+115	+134	+153	+172
B	+21	+40	+59	+78	+97	+116	+135	+154	+173
C	+22	+41	+60	+79	+98	+117	+136	+155	+174
D	+23	+42	+61	+80	+99	+118	+137	+156	+175
E	+24	+43	+62	+81	+100	+119	+138	+157	+176

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value +28 is encoded as the string “AY”.

Table B.3-2 *(continued)*

Second character of the encoded value	First character of the encoded value								
	A	B	C	D	E	F	G	H	Y
F	+25	+44	+63	+82	+101	+120	+139	+158	+177
G	+26	+45	+64	+83	+102	+121	+140	+159	+178
H	+27	+46	+65	+84	+103	+122	+141	+160	+179
Y	+28	+47	+66	+85	+104	+123	+142	+161	+180

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value +28 is encoded as the string “AY”.

B.4 Encoding returned by the EncodeDecimal algorithm

Table B.4-1 — EncodeDecimal encoding in base-32 (0 to 351)

Second character of the encoded value	First character of the encoded value									
	0	1	2	3	4	5	6	7	8	9	A
0	0	32	64	96	128	160	192	224	256	288	320
1	1	33	65	97	129	161	193	225	257	289	321
2	2	34	66	98	130	162	194	226	258	290	322
3	3	35	67	99	131	163	195	227	259	291	323
4	4	36	68	100	132	164	196	228	260	292	324
5	5	37	69	101	133	165	197	229	261	293	325
6	6	38	70	102	134	166	198	230	262	294	326
7	7	39	71	103	135	167	199	231	263	295	327
8	8	40	72	104	136	168	200	232	264	296	328
9	9	41	73	105	137	169	201	233	265	297	329
A	10	42	74	106	138	170	202	234	266	298	330
B	11	43	75	107	139	171	203	235	267	299	331
C	12	44	76	108	140	172	204	236	268	300	332
D	13	45	77	109	141	173	205	237	269	301	333
E	14	46	78	110	142	174	206	238	270	302	334
F	15	47	79	111	143	175	207	239	271	303	335
G	16	48	80	112	144	176	208	240	272	304	336
H	17	49	81	113	145	177	209	241	273	305	337
Y	18	50	82	114	146	178	210	242	274	306	338
J	19	51	83	115	147	179	211	243	275	307	339
K	20	52	84	116	148	180	212	244	276	308	340
L	21	53	85	117	149	181	213	245	277	309	341
M	22	54	86	118	150	182	214	246	278	310	342
N	23	55	87	119	151	183	215	247	279	311	343
Z	24	56	88	120	152	184	216	248	280	312	344
P	25	57	89	121	153	185	217	249	281	313	345
Q	26	58	90	122	154	186	218	250	282	314	346
R	27	59	91	123	155	187	219	251	283	315	347
S	28	60	92	124	156	188	220	252	284	316	348

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value 31 is encoded as the string “0V”.

Table B.4-1 *(continued)*

Second character of the encoded value	First character of the encoded value										
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**
T	29	61	93	125	157	189	221	253	285	317	349
U	30	62	94	126	158	190	222	254	286	318	350
V	31	63	95	127	159	191	223	255	287	319	351

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value 31 is encoded as the string “0V”.

Table B.4-2 — EncodeDecimal encoding in base-32 (352 to 703)

Second character of the encoded value	First character of the encoded value										
	B	**C**	**D**	**E**	**F**	**G**	**H**	**Y**	**J**	**K**	**L**
0	352	384	416	448	480	512	544	576	608	640	672
1	353	385	417	449	481	513	545	577	609	641	673
2	354	386	418	450	482	514	546	578	610	642	674
3	355	387	419	451	483	515	547	579	611	643	675
4	356	388	420	452	484	516	548	580	612	644	676
5	357	389	421	453	485	517	549	581	613	645	677
6	358	390	422	454	486	518	550	582	614	646	678
7	359	391	423	455	487	519	551	583	615	647	679
8	360	392	424	456	488	520	552	584	616	648	680
9	361	393	425	457	489	521	553	585	617	649	681
A	362	394	426	458	490	522	554	586	618	650	682
B	363	395	427	459	491	523	555	587	619	651	683
C	364	396	428	460	492	524	556	588	620	652	684
D	365	397	429	461	493	525	557	589	621	653	685
E	366	398	430	462	494	526	558	590	622	654	686
F	367	399	431	463	495	527	559	591	623	655	687
G	368	400	432	464	496	528	560	592	624	656	688
H	369	401	433	465	497	529	561	593	625	657	689
Y	370	402	434	466	498	530	562	594	626	658	690
J	371	403	435	467	499	531	563	595	627	659	691
K	372	404	436	468	500	532	564	596	628	660	692
L	373	405	437	469	501	533	565	597	629	661	693
M	374	406	438	470	502	534	566	598	630	662	694
N	375	407	439	471	503	535	567	599	631	663	695
Z	376	408	440	472	504	536	568	600	632	664	696
P	377	409	441	473	505	537	569	601	633	665	697
Q	378	410	442	474	506	538	570	602	634	666	698
R	379	411	443	475	507	539	571	603	635	667	699
S	380	412	444	476	508	540	572	604	636	668	700
T	381	413	445	477	509	541	573	605	637	669	701
U	382	414	446	478	510	542	574	606	638	670	702
V	383	415	447	479	511	543	575	607	639	671	703

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value 383 is encoded as the string “BV”.

Table B.4-3 — EncodeDecimal encoding in base-32 (704 to 999)

Second character of the encoded value	First character of the encoded value									
	M	**N**	**Z**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**
0	704	736	768	800	832	864	896	928	960	992
1	705	737	769	801	833	865	897	929	961	993
2	706	738	770	802	834	866	898	930	962	994
3	707	739	771	803	835	867	899	931	963	995
4	708	740	772	804	836	868	900	932	964	996
5	709	741	773	805	837	869	901	933	965	997
6	710	742	774	806	838	870	902	934	966	998
7	711	743	775	807	839	871	903	935	967	999
8	712	744	776	808	840	872	904	936	968	—
9	713	745	777	809	841	873	905	937	969	—
A	714	746	778	810	842	874	906	938	970	—
B	715	747	779	811	843	875	907	939	971	—
C	716	748	780	812	844	876	908	940	972	—
D	717	749	781	813	845	877	909	941	973	—
E	718	750	782	814	846	878	910	942	974	—
F	719	751	783	815	847	879	911	943	975	—
G	720	752	784	816	848	880	912	944	976	—
H	721	753	785	817	849	881	913	945	977	—
Y	722	754	786	818	850	882	914	946	978	—
J	723	755	787	819	851	883	915	947	979	—
K	724	756	788	820	852	884	916	948	980	—
L	725	757	789	821	853	885	917	949	981	—
M	726	758	790	822	854	886	918	950	982	—
N	727	759	791	823	855	887	919	951	983	—
Z	728	760	792	824	856	888	920	952	984	—
P	729	761	793	825	857	889	921	953	985	—
Q	730	762	794	826	858	890	922	954	986	—
R	731	763	795	827	859	891	923	955	987	—
S	732	764	796	828	860	892	924	956	988	—
T	733	765	797	829	861	893	925	957	989	—
U	734	766	798	830	862	894	926	958	990	—
V	735	767	799	831	863	895	927	959	991	—

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value 735 is encoded as the string “MV”.

B.5 Encoding returned by the EncodeStoreyBase34 algorithm

Table B.5-1 — EncodeStoreyBase34 encoding in base-34 (0 to 407)

Second character of the encoded value	First character of the encoded value											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**
0	0	34	68	102	136	170	204	238	272	306	340	374
1	1	35	69	103	137	171	205	239	273	307	341	375
2	2	36	70	104	138	172	206	240	274	308	342	376
3	3	37	71	105	139	173	207	241	275	309	343	377
4	4	38	72	106	140	174	208	242	276	310	344	378
5	5	39	73	107	141	175	209	243	277	311	345	379
6	6	40	74	108	142	176	210	244	278	312	346	380
7	7	41	75	109	143	177	211	245	279	313	347	381
8	8	42	76	110	144	178	212	246	280	314	348	382
9	9	43	77	111	145	179	213	247	281	315	349	383
A	10	44	78	112	146	180	214	248	282	316	350	384
B	11	45	79	113	147	181	215	249	283	317	351	385
C	12	46	80	114	148	182	216	250	284	318	352	386
D	13	47	81	115	149	183	217	251	285	319	353	387
E	14	48	82	116	150	184	218	252	286	320	354	388
F	15	49	83	117	151	185	219	253	287	321	355	389
G	16	50	84	118	152	186	220	254	288	322	356	390
H	17	51	85	119	153	187	221	255	289	323	357	391
Y	18	52	86	120	154	188	222	256	290	324	358	392
J	19	53	87	121	155	189	223	257	291	325	359	393
K	20	54	88	122	156	190	224	258	292	326	360	394
L	21	55	89	123	157	191	225	259	293	327	361	395
M	22	56	90	124	158	192	226	260	294	328	362	396
N	23	57	91	125	159	193	227	261	295	329	363	397
Z	24	58	92	126	160	194	228	262	296	330	364	398
P	25	59	93	127	161	195	229	263	297	331	365	399
Q	26	60	94	128	162	196	230	264	298	332	366	400
R	27	61	95	129	163	197	231	265	299	333	367	401
S	28	62	96	130	164	198	232	266	300	334	368	402
T	29	63	97	131	165	199	233	267	301	335	369	403
U	30	64	98	132	166	200	234	268	302	336	370	404
V	31	65	99	133	167	201	235	269	303	337	371	405
W	32	66	100	134	168	202	236	270	304	338	372	406
X	33	67	101	135	169	203	237	271	305	339	373	407

NOTE Each cell in the main body of the table contains the value to be encoded. The cell column (first character) and row (second character) indicate the encoded value, i.e. the numeric value 33 is encoded as the string “0X”.