### 4.6.3 Usage of @precision and @decimals attributes

A Numeric Item MUST have either a @decimals attribute or a @precision  attribute unless it is of the fractionItemType or of a type that is derived by restriction from fractionItemType or has a nil value, in which case, it MUST NOT have either a @decimals attribute or a @precision attribute.
A Numeric Item MUST NOT have both a @decimals attribute and a @precision attribute.
A Non-Numeric Item MUST NOT have either a @decimals or a @precision attribute.
When determining whether two Numeric Items are V-Equal (a predicate that is used in the definition of various other equality type predicates) it is necessary to take into consideration the values of @ decimals (or the decimals inferred from the value of the @ precision attribute) for the two numeric items. The formal definition of V-Equal for two numeric items is given in Section 4.10.

### 4.6.4 The @decimals attribute (optional)

The @decimals attribute MUST be an integer or the value "INF" that specifies the number of decimal places to which the value of the fact represented may be considered accurate, possibly as a result of rounding or truncation. If a numeric fact has a @decimals attribute with the value "n" then it is known to be correct to "n" decimal places. (See Section 4.6.7.2 for the normative definition of 'correct to "n" decimal places').
The meaning of decimals="INF" is that the lexical representation of the number is the exact value of the fact being represented.

NOTE: The definitions in this specification mean that @decimals and by inference, @precision indicate the range in which the actual value of the fact that gave rise to its expressed value in the XBRL Instance lies.

EXAMPLE 11: DECIMALS AND LEXICAL REPRESENTATION

| Example | Meaning |
|---|---|
| decimals="2" | The value of the numeric fact is known to be correct to 2 decimal places. |
| decimals="-2" | The value of the numeric fact is known to be correct to -2 decimal places, i.e. all digits to the left of the hundreds digit are accurate. |

The simple type decimalsType defines the legal values for the @decimals attribute. Its XML Schema definition is as follows:

### 4.6.5 The @precision attribute (optional)

The @precision attribute MUST be a non-negative integer or the string "INF" that conveys the arithmetic precision of a measurement, and, therefore, the utility of that measurement to further calculations. Different software packages may claim different levels of accuracy for the numbers they produce. The @precision attribute allows any producer to state the precision of the output in the same way. If a numeric fact has a @precision attribute that has the value "n" then it is correct to "n" significant figures (see Section 4.6.1 for the normative definition of 'correct to "n" significant figures'). An application SHOULD ignore (i.e. replace with zeroes) any digits after the first "n" decimal digits, counting from the left, starting at the first non-zero digit in the lexical representation of any number for which the value of precision is specified or inferred to be n.

The meaning of precision="INF" is that the lexical representation of the number is the exact value of the fact being represented.

EXAMPLE 12: PRECISION AND LEXICAL REPRESENTATION

| Example | Meaning |
|---|---|
| precision="9" | Precision of nine digits. The first 9 digits, counting from the left, starting at the first non-zero digit in the lexical representation of the value of the numeric fact are known to be trustworthy for the purposes of computations to be performed using that numeric fact. |

The simple type precisionType has been provided to define the value space for the value of the @precision attribute. Its definition is as follows:

### 4.6.6 Inferring decimals

The following rules enable XBRL Instance consumers to infer a value for the @decimalsattribute of a Numeric Item when none is supplied.

For a Numeric Item of type fractionItemType or type derived by restriction from fractionItemType, a consuming application MUST infer the precision to be equal to 'INF' if it is to be used in calculations.

If, on a Numeric Item, the @precision attribute is present rather than the @decimals attribute, then a consuming application MUST infer the decimals of that numeric fact if it is to be used in calculations or searches for duplicates in XBRL Instances.

If the value of the @precision attribute of a Numeric Item is equal to 0 , nothing is known about the precision of the number, nothing can be inferred about decimals, and thus any consuming V-Equals comparison must be false, and any calculation link summation involving the item must be inconsistent.

If the value of the @precision attribute is INF then the inferred decimals value is INF.

If the value of the @precision attribute is not INF and greater than 0 then the decimals value is

For an item of numeric value 0, the inferred decimals is deemed to be INF, treating data values of zero as a singularity of infinite decimals accuracy (regardless of non-zero value of @precision attribute or item syntax, e.g., 0, or 000, or .00).

Otherwise the inferred decimals is given by the following expression;

decimals = precision - int(floor(log10(abs(number(item))))) – 1

where

| | |
|---|---|
| precision | is the value of the @precision attribute, |
| int( ) | is a function returning an integer of its argument, |
| floor( ) | is a function returning the largest integer less than or equal to its argument, |
| log10( ) | is a function returning the logarithm base 10 of its argument, |
| abs( ) | is a function returning the absolute value of its argument, |

number( ) is a function providing a numeric conversion if its argument is not internally numeric (as may be needed for the math computations), and item is the item's value (PSVI typed numeric node value if available, or otherwise inner text of numeric item node).

EXAMPLE 13: LEXICAL REPRESENTATION, PRECISION AND DECIMALS

| Lexical Representation | Value of the precision attribute | Inferred value of the @decimals attribute |
|---|---|---|
| 123 | 5 | 2 = 5 - int(floor(log10(abs(number(123))))) - 1 |
| 123.4567 | 5 | 2 = 5 - int(floor(log10(abs(number(123.4567))))) - 1 |
| 12300 | 3 | -2 = 3 - int(floor(log10(abs(number(12.300))))) - 1 |
| 12345 | 3 | -2 = 3 - int(floor(log10(abs(number(12.345))))) - 1 |
| 0.150 | 1 | 1 = 1 - int(floor(log10(abs(number(0.150))))) - 1 |
| 0.095 | 1 | 2 = 1 - int(floor(log10(abs(number(0.095))))) - 1 |
| 0.1 | 0 | nothing can be inferred |

### 4.6.7 Rounding of numbers (Definitions pertaining to accuracy)

The following definitions are provided for clarity regarding accuracy-related features of this specification, i.e. @decimals and @precision attributes.

If the lexical representation of the value of a number is said to be correct to n significant figures it means that the first "n" decimal digits, counting from the left, starting at the first non-zero digit in the lexical representation of the number are known to be accurate for the purposes of computations to be performed using that number. (Note: in the following it is assumed that all zeros to the left of the decimal point and to the left of the first non-zero digit in the decimal representation have been removed first).

In the decimal representation of a number, a significant figure is any one of the digits 1, 2, 3...9 that specify the magnitude of a number. Zero (0) is a significant figure except when it appears to the left of all non-zero digits or is used solely to fill the places of unknown or discarded digits (after truncation or rounding - see later). Thus, in the number "0.00263", there are three significant figures: 2, 6, and 3. The zeroes are not significant. In the number "3809" all four of the digits are significant. In the number "46300" the digits 4, 6, and 3 are known to be significant but it is not possible to conclude anything concerning the two zeroes as they are written. This ambiguity can be removed by writing the number in terms of powers of ten. If there are three significant figures the representation becomes 4.63 × 104; if there are four significant figures it becomes 4.630 × 104, etc.

#### 4.6.7.1 "Rounding" and "Truncation"

Rounding of numbers MUST be performed as specified in [ISO] (ISO80000-1, Annex B Rounding of numbers). Rounding means replacing the magnitude of an original number by another number called the rounded number, selected from the sequence of integral multiples of a chosen rounding range. [IEEE] (IEEE 4.3 Rounding-direction) lists the rounding-direction attribute affects all computational operations that might be inexact.

The first rounding is “Rounding”, and its rounding-direction is “roundTiesToEven”.
If there is only one integral multiple nearest the original number, then that is accepted as the rounded number.

If there are two successive integral multiples equally near the original number. the even multiple is selected as the rounded number. This rule is of special advantage when treating. for example. series of measurements in such a way that the rounding errors are minimized.

The second rounding is “Truncate”, and its rounding-direction is “roundTowardZzero”. An integral multiple closest to and no greater in multitude than infinity precise result the original number is accepted as the truncated number.

Rounding in more than one stage by the application of the rules given above may lead to errors. Therefore, the rounding MUST always be carried out in only one step.

EXAMPLE 12,254 should be rounded to 12,3 and not first to 12,25 and then to 12,2.

The rules given above should be used only if no special criteria for the selection of the rounded number MUST be taken into account. For instance, in case where safety requirements or other limits MUST be respected, it is advisable to round only in one direction.

| original number | rounding range | integral multiple | Rounded number (roundTiesToEven) | Truncated number (roundTowardZzero) |
|---|---|---|---|---|
| -12.251 | 0.1 | -12.4; -12.3, -12.2; -12.1; etc. | -12.3 | -12.2 |
| -12.233 | | | -12.2 | |
| 12.233 | | 12.1; 12.2; 12.3; 12.4; etc. | 12.2 | 12.2 |
| 12.250 | | | | |
| 12.275 | | | 12.3 | |
| -1 225.1 | 10 | -1 240; -1 230; -1 220; -1210; etc. | -1 230 | -1 220 |
| -1 223.3 | | | -1 220 | |
| 1 223.3 | | 1 210; 1 220; 1 230; 1 240; etc. | 1 220 | 1 220 |
| 1 225.0 | | | | |
| 1 227.5 | | | 1 230 | |

### 4.6.7.2 "Correct to *n* Decimal Places"

If the representation of a number is correct to n decimal places then rounding range is given by the following expression;

$$\text{rounding range} = (1/10)^{@decimals}$$

EXAMPLE 14: ROUNDING

| Original number | @decimals=n | | | | |
|---|---|---|---|---|---|
| | *n=-3* | *n=-1* | *n=0* | *n=1* | *n=2* |
| | rounding range = 1000 | rounding range = 10 | rounding range = 1 | rounding range = 0.1 | rounding range = 0.01 |
| 35 643 | 36 000 | 35 640 | 3 5643 | 35 643.0 | 35 643.00 |
| 3.5643 | 0 | 0 | 4 | 3.6 | 3.56 |
| 3.5673 | 0 | 0 | 4 | 3.6 | 3.57 |
| 0.49787 | 0 | 0 | 0 | 0.5 | 0.50 |

| | | | | | |
|---|---|---|---|---|---|
| 3.9999 | 0 | 0 | 4 | 4.0 | 4.00 |
| 9.999991 | 0 | 10 | 10 | 10.0 | 10.00 |
| 22.55 | 0 | 20 | 23 | 22.6 | 22.6 |
| 22.65 | 0 | 20 | 23 | 22.6 | 22.6 |
| 0.0019 | 0 | 0 | 0 | 0.0 | 0.00 |
| 0.00002 | 0 | 0 | 0 | 0.0 | 0.00 |

The same procedure MAY be followed for any value of n, and we then say that a particular lexical representation of the value of a number is correct to n decimal places. It is possible that this technique has been used to create the lexical representation of a fact in an XBRL Instance with a @decimals attribute of n.

### 4.6.7.3 "Correct to *n* Significant Figures"

If the representation of a number is correct to n significant figures then first infer @decimals from @precision with method in 4.6.6, then calculates rounding range from the equation defined in 4.6.7.1.

EXAMPLE 15: ROUNDING

| Original value | @precision=n | | | | | |
|---|---|---|---|---|---|---|
| | *n=1* | rounding range | *n=2* | rounding range | *n=3* | rounding range |
| 3.5643 | 4 | 1 | 3.6 | 0.1 | 3.56 | 0.01 |
| 3.5673 | 4 | 1 | 3.6 | 0.1 | 3.57 | 0.01 |
| 0.49787 | 0.5 | 0.1 | 0.50 | 0.01 | 0.498 | 0.001 |
| 3.9999 | 4 | 1 | 4.0 | 0.1 | 4.00 | 0.01 |
| 9.999991 | 10 | 10 | 10 | 1 | 10.0 | 0.1 |
| 22.55 | 20 | 10 | 23 | 1 | 22.6 | 0.1 |
| 22.65 | 20 | 10 | 23 | 1 | 22.6 | 0.1 |
| 0.0019 | 0.002 | 0.001 | 0.0019 | 0.0001 | 0.00190 | 0.00001 |
| 0.00002 | 0.00002 | 0.00001 | 0.000020 | 0.000001 | 0.0000200 | 0.0000001 |

The same procedure MAY be followed for any value of n, and we then say that a particular lexical representation of the value of a number is correct to n significant figures. It is possible that this technique has been used to create the lexical representation of a fact in an XBRL Instance with a @precision attribute of n.

6. Reference

ISO 4217 Currency code
(see https://www.iso.org/iso-4217-currency-codes.html)
ISO 639 Language codes
(see https://www.iso.org/iso-639-language-codes.html)
ISO 3166 Codes for the representation of names of countries and their subdivisions
(see https://www.iso.org/iso-3166-country-codes.html)
ISO 8601 Date and time — Representations for information interchange
(see https://www.iso.org/iso-8601-date-and-time-format.html)
ISO 80000-1 Quantities and units— Part 1: General
(see https://www.iso.org/standard/30669.html)