*The standard does not say which of these results is "correct". Rather than change the implementation, it might be better to document that such code must be avoided if reproducible behavior is required.*]

**14.4. Interchange representations and encodings.** The purpose of interchange representations and encodings is to allow the loss-free exchange of Level 2 interval data between 754-conforming implementations. This is done by imposing a standard Level 3 representation using Level 2 number datums and delegating interchange encoding of number datums to the IEEE 754 standard.

Let $\boldsymbol{x}$ be a datum of the bare interval inf-sup type $\mathbb{T}$ derived from a supported 754 format $\mathbb{F}$. Its standard Level 3 representative is an ordered pair $(\texttt{inf}(x), \texttt{sup}(x))$ of two Level 2 $\mathbb{F}$-numbers as defined in §12.12.8. For example, the only representative of Empty is the pair $(+\infty, -\infty)$ and the only representative of $[0, 0]$ is the pair $(-0, +0)$.

Let $\boldsymbol{x}_{dx}$ be a datum of the decorated interval type $\mathbb{DT}$ derived from $\mathbb{T}$. Its standard Level 3 representative is an ordered triple $(\texttt{inf}(\boldsymbol{x}), \texttt{sup}(\boldsymbol{x}), dx)$ of two Level 2 $\mathbb{F}$-datums and a decoration. For example, the only representative of $\text{Empty}_{\texttt{trv}}$ is the triple $(+\infty, -\infty, \texttt{trv})$ and the only representative of NaI is the triple $(\text{NaN}, \text{NaN}, \texttt{ill})$.

Interchange Level 4 encoding of an interval datum is a bit string that comprises, in the order defined above, the 754-2008 interchange encodings of the two floating-datums, and, for decorated intervals, of the decoration represented as an **octet** (bit string of length 8, equivalently 8-bit byte) as follows:

| | |
|---|---|
| `ill` | 00000000 |
| `trv` | 00000100 |
| `def` | 00001000 |
| `dac` | 00001100 |
| `com` | 00010000 |

This encoding permits future refinement without disturbing the propagation order of the decorations.

[*Note. The above rules imply that an interval has a unique interchange representation if it is not* NaI *and in a binary format, but not generally otherwise. The reason for the rules is that the sign of a zero bound cannot convey any information relevant to intervals; but an implementation may potentially use cohort information, or a* NaN *payload.*]

When (optional) compressed intervals are supported, their interchange representation is as described above for bare intervals for non-empty intervals, but decorations are represented as a pair of floating-point datums, the first of which is a generic NaN (no payload information), and the second is a floating-point integer that encodes the decoration as follows:

| | |
|---|---|
| `ill` | 0.0 |
| `emp` | 2.0 |
| `trv` | 4.0 |
| `def` | 8.0 |
| `dac` | 12.0 |

[*Note. These encodings match the octet-encoded decorations of decorated intervals, when interpreted as small integers, with a new decoration* `emp` *to represent a compressed* Empty *interval. The* `com` *decoration cannot occur in a compressed interval.*]

Export and import of interchange formats normally occurs as a sequence of octets, e.g. in a file or a network packet. There is therefore a need to define the **octet-encoding** that maps the conceptual Level 4 bit string encodings of floating-point datums (as specified by 754-2008) and of decorations (not specified in 754-2008) into an octet sequence. Octet-encoding might optionally insert padding zero octets for alignment.

Applications exchanging data need to describe the types and layout thereof; standards like this one (and 754-2008) only define the representation of individual datums, and then only as conceptual Level 4 entities. Environments whose primary focus is universal portability (e.g. Java) may fully define the representation at the level of an octet sequence, even when this does not match the natural in-memory layout of the platform. The standard merely requires the parameters of the