Comments from Japan, Netherlands and United Kingdom received on the FCD 11404 ballot (May 2005).

National Body Comments

Japan

Japan disapproves the FCD document. We found so many errors in the document, and we cannot approve it in its current form. Most of these errors are editorial, but some of them have technical significance. We are ready to change our vote to "approve" if these errors are corrected.

```
<< Foreword >>

This foreword seems to be non-conforming to the standard template of
forewords for JTC1 standards.  The standard template begins with the
paragraph like "ISO ( ... ) and IEC ( ... ) form the specialized
system ... ISO and IEC have established a joint technical committee,
ISO/IEC JTC 1.".  In contrast to this, this foreword speaks about ISO
only.  There are slight differences also in the third paragraph:
"technical committees" -> "the joint technical committee", "member body"
-> "national body".

<< 2 >>

  ISO/IEC 10646-1:2000, Information technology --- Universal Multiple-Octet
  Coded Character Set (UCS) ---
  Part 1: Architecture and Basic Multilingual Plane

There are newer versions of ISO/IEC 10646-1.  The latest version
should be cited.

<< 3.33 >>

  <<EBNF>> name of a non-terminal symbol [ISO/IEC 14977]

The notation "<<EBNF>>" is used without any explanations.
```

```
<< 3.50, Note >>

  NOTE In this context, the term "sign" is used in its terminological
  sense (e.g., a symbol) and not in its mathematical
  sense (e.g., positive of negative).

"positive of negative" -> "positive or negative".

<< 3.60 >>

  value U such that, for all values s in the value space in a datatype
  which is bounded below, s <= U

"bounded below" -> "bounded above".

<< 5.1, Table 5-1 >>

  ( ) left parentheses, right parenthesis start/end group symbols

"parentheses" -> "parenthesis".

<< 5.2, second bullet >>

  A reference to a non-terminal symbol syntactic object consists of the
  non-terminal-symbol in fix width italic
  courier, e.g. type-declaration.

"fix width" -> "fixed width".

<< 6.2 >>

There are two Notes in this section.  They should be numbered.

<< 6.2, second Note >>

  NOTE A numeric datatype, which includes characterizing operations such
  as IsEqual and InOrder, may include

The operation "IsEqual" has been renamed to "Equal".

<< 6.3.5 >>

  Let M be the mathematical datatype and C be the corresponding
```

computational datatype, and let P be

We think that the term "mathematical datatype" is not appropriate here. It is nothing other than the type of real or complex numbers, so should be rephrased more explicitly.

<< 6.3.5, Note >>

NOTE The computational model described above allows a mathematically dense datatype to be mapped to a
datatype with fixed-length representations and nonetheless evince intuitively acceptable mathematical behavior.

The form of the verb "evince" seems incorrect. Should it be changed to "to evince", or "evinces"?

<< 6.6, Note 5 >>

NOTE 5 IsEqual is always a characterizing operation on datatypes with the equality property.

The operation "IsEqual" has been renamed to "Equal".

According to 6.3.1, every value space has a notion of equality.
Is there any type without the equality property?

<< 6.8.8, second bullet >>

conceptually semi-structured, have either the component datatypes or the access method specified, but

The word "have" should be changed to "having" which is used in the first and the third bullets.

<< 6.9 >>

That is, a normative datatype does not have a specific value space, but it may specify properties that any
conforming value space must have. Similarly, a normative datatype may specify operations that must be
supported by a conforming datatype, without that set of operations itself being sufficient to characterize any
one datatype.

For example, the normative datatype Any can be satisfied by any GPD
datatype, with any value space. The
only requirement is that Equal is defined on the value space."

We cannot understand these two paragraphs at all. They may be misplaced
or some pieces of text may be missing.

The double-quote character at the end of the second paragraph should
be deleted.

<< 6.9 >>

This International Standard contains many provisions. Some provisions
apply to datatypes in general, e.g., a
datatype consists of a value space, properties, and characterizing
operations --- a "statement" provision.
Some provisions apply to specific datatypes, e.g., a mapping to the GPD
integer datatype shall be a
datatype that is numeric --- a "requirement" provision.

We cannot understand why the terms "statement provision" and
"requirement provision" are mentioned here. Their definitions are
given in 3.46, 3.51 and 3.56. It would be better to remove them.

Declarations may contain provisions describe via
annotations (outside the scope of this International Standard).

The form of the verb "describe" seems incorrect.

<< 7.1 >>

```
non-quote-character = letter |
digit |
underscore |
special |
apostrophe |
space ;
```

The order of "underscore" and "special" is not consistent with the
order of their syntax definitions. The same comment also applies to
8.1.4 and 10.1.5.

<< 7.3.1 >>

```
  pseudo-letter-like = letter |
  digit |
  underscore ;
```

The first two alternatives seem incorrect. "letter" should be changed to "letter-like" and "digit" to "digit-like". According to the current syntax, extended characters are allowed only as the first character of identifiers. This is curious. The non-terminal "digit-like" is not referred to.

<< 7.6 >>

```
  program-statement = type-specifier |
  declaration |
  normative-datatype-definition ;
```

"normative-datatype-definition" is not a correct non-terminal name. It should be "normative-datatype-declaration" according to 9.4.

<< 8.1.2 >>

```
  list-source-reference = identifier |
  '"', URI-text, '"" ;
  URI-text = '"', URI defined by IETF RFC2396, '"" ;
```

The terminal "double-quote" is inappropriately denoted. It should be "single quote - double quote - single quote", but is "single - double - double" at the end of "list-source-reference" and at the end of "URI-text".

According to this syntax, a list-source-reference must have two double-quotes preceding and following it. Is this intended?

> shall denote a valid value of the URI datatype. When the list-source is an objectidentifier-value, it shall denote

There is no reference to "objectidentifier-value" in the syntax of "list-source-reference".

> NOTE 1 Other uses of the IDN syntax make stronger requirements on the uniqueness of state-literal identifiers.

This section has only one Note. It should not be numbered.

<< 8.1.2, Example 2 >>

```
  EXAMPLE 2 Enumerated types {short, medium, tall} and {light, medium,
  heavy} are distinct types of the family
  "enumerated", even though they have exactly the same number of
  elements, and the same characterizing operations:
  IsEqual and InOrder. Enumerated types {short, medium, tall} and {short,
  moderate, medium, tall} are distinct types.
  It is outside the scope of this International Standard whether or not
  the value medium is the same in both enumerated
  types.
```

This whole example is misplaced. It is an example of "enumerated type"
and thus should be moved to 8.1.3.

The operation "IsEqual" has been renamed to "Equal".

<< 8.1.3 >>

```
  enumerated-value = enumerated-value-list |
  URI-to-value-space ;
```

There is no syntax rule for "URI-to-value-space".

<< 8.1.3 >>

```
  these values is given by the sequence of their occurrence in the
  enumerated-value-list, designated the
  naming sequence.
```

The phrase "designated the naming sequence" was hard to understand
for us. We think that the phrase should be made more explict, such
as "in the enumerated-value-list, which shall be referred to as the
naming sequence of the enumerated datatype".

<< 8.1.3, Note 2 >>

```
  NOTE 2 The ordering on enumeration types imposed by programming
  languages is a convenience that allows
  programs to reference all the values via for-loops and enables the
  compiler to use integer encodings to simplify
  implementation. Properly, the Enumeration type should be chosen over
  the State type only when the ordering has
```

The term "enumeration type" is not correct. It should be changed to
"enumerated type" (two places).

<< 8.1.4, Example >>

```
  type Latin1 = character({ iso standard 8859 1 });
```

The word "part" should be inserted between "8859" and "1", in order
to conform to the preceding example.

<< 8.1.6 >>

```
  the resolution is to one
  radix(-factor) of the specified time-unit. time-unit, and radix and
```

The formula is misformatted. It should be "radix raised to the
power (-factor)".

```
  InOrder(x, y: time(time-unit, radix, factor)): boolean is true if the
  point in time designated by x precedes
  that designated by y; else false.
```

"InOrder" is a "less-than-or-equal-to" operation. According to this
definition, it looks like a "less-than" operation.

<< 8.1.7 >>

```
  NonNegative(x: integer): boolean is
  true if x = 0 or x can be developed by one or more iterations of adding
  1,
```

"adding to what" is not mentioned. "to 0" should be inserted after
"adding 1".

```
  Quotient(x, y: integer): integer, where 0 < y, is the upperbound of the
  set of all integers z such that
```

The notation of operation definitions given in 8.1 does not allow the
insertion of "where" condition just after the operation signature

(just before "is" or "="). Probably, the definition in 8.1 should
be made more precise.

<< 8.1.8 >>

```
  value 0. The rational value denoted by the form signed-number is:
  Promote(signed-number),
  and the rational value denoted by the form signed-number/number is:
```

The fonts of "form signed-number" and of "form signed-number/number"
are not consistent.

```
  Operations: Equal, NonNegative, InOrder, Negate, Add, Multiply,
  Reciprocal, Promote.
```

The order of operation names is not consistent with the order of
the following operation definitions.

<< 8.1.9 >>

```
  InOrder(x,y: scaled (r,f)): boolean = rational.InOrder(x,y)
```

Here, x and y are values of a scaled type. Is it possible to
apply operations of the rational type to them? The value space
of a scaled type is a subset of the rational type, but nevertheless
they are different types.

<< 8.1.9, Note 2 >>

```
  NOTE 2 Any reasonable rounding algorithm is equally acceptable. What is
  required is that any rational value v which
  is not a value of the scaled datatype is mapped into one of the two
  scaled values n.r^(-f) and (n+1).r^(-f), such that in the
  Rational value space, n.r^(-f) < v < (n+1).r^(-f).
```

We suspect that this Note is not correct. According to the definition
of "Round" given in this section, rounding should be toward zero
for positive values.

<< 8.1.10 >>

```
  Let . .denote the mathematical real value space and for v in ., let | v
  | denote the absolute value of v. Let V
```

The real value space is denoted by a strange symbol, looking like
a vertical stroke. It should be given an appropriate name, probably
"R" in a special font.

<< 8.1.10, third bullet >>

```
  for each r in . . such that | r | < epsilon, there exists at
  least one r' in V such that | r - r' | <= epsilon2;.
```

The sequence "semicolon-period" is strange. The semicolon should be
deleted.

<< 8.1.11, third bullet >>

```
  for each v in C such that | v | < epsilon, there exists at least one v in V
  such that | v - v | _ _ _;.
```

The formula is obviously incomplete. Three boxes appear at the end of
the formula. "v-v" should be changed to "v-v'".

The semicolon should be deleted.

<< 8.2.2 >>

  Properties: The subtype is bounded (above, below, both) if the base
  datatype is so bounded or if no selectrange
  appears in the select-list or if all select-ranges in the select-list
  specify the corresponding
  bounds.

The condition is not well-stated, and we cannot understand what
is the intended condition. Probably, "or if" should be changed to
"and".

<< 8.2.4 >>

```
  maximum-size = value-expression |
  "*" ;
  minimum-size = value-expression ;
```

The definition of "minimum-size" should be given before that of
"maximum-size".

```
  Subtypes: Any size subtype of the same base datatype, such that
  base-minimum-size <=  subtypeminimum-size, and
  subtype-maximum-size <= base-maximum-size
```

There is an inappropriate line break in this sentence.

The sentence is not terminated properly.

The header "Subtypes" is not mentioned in the template definitions in 8.2.

<< 8.2.6 >>

```
  NOTE The value space of a datatype is the set of values specified in
  the definition of the datatype. Sentinel
```

This section contains two more Notes at the end of the section. This Note should also be given a number.

```
  IsEqual from values in the value space. Sentinel values must be
```

The operation "IsEqual" has been renamed to "Equal".

<< 8.3.1 >>

```
  alternative-list = alternative, { ",", alternative },
  [ default-alternative ] ;
```

Is this syntax definition correct? We suspect that there should be a comma just before "default-alternative".

```
  alternative = tag-value-list, [ field-identifier ], ":",
  alternative-type ;
```

The second line of this definition should be indented appropriately.

```
  value-expression. A select-item which is a select-range specifies all
  values v of the tag datatype such
```

The font of "select-item" is obviously inappropriate (too thin).

```
  if Discriminant(x) and Discrminant(y) select the same alternative, then
  type.Equal(Cast.type(x), Cast.type(y)),
```

"Discrminant" -> "Discriminant".

<< 8.3.3 >>

```
  Aj = E1j x E2j x ... x Emjj,
```

"Emjj" -> "Emj".

```
  p: I1 x I2 x ... x In _ ST x (A0 | A1 | A2 | ... | AN).
```

The formula is incomplete. There is a box symbol in the formula.

```
  datatype many contain many distinct values which differ in their
```

"many" -> "may" (first occurrence).

<< 8.3.3, Note 6 >>

```
  the tag datatype is the unspecified state datatype and each
  alternative, including "normal", has the form:
```

Shouldn't this "normal" be changed to "*normal"?

<< 8.4.1 >>

```
  Operations: Equal, FieldSelect, Aggregate.
```

There is no "Aggregate" operation for this type. It should be replaced
by "FieldReplace".

```
  FieldReplace.field-identifier(x: record (field-list), y: field-type):
  record (field-list) is that value z:
  record(field-list) such that FieldSelect.field-identifier(z) = y, and
  for all other fields f in record(fieldlist),
  FieldSelect.f(x) = FieldSelect.f(z)
```

The first line should not be indented.

<< 8.4.2 >>

Syntax rules should be aligned properly at the position of equality symbols.

```
  attribute = { override-qualifier }, attribute-identifier, ":",
  attribute-type ;
```

Is this syntax rule intended? It allows repeated appearances of
the word "override", which makes no sense, we think.

```
  Description: .
  Syntax:
  override-qualifier = "override" ;
```

These three lines should be deleted.

  Components: override may be used with a class attribute definition.
  Components: A list of attributes, each of which associates a
  attribute-identifier with a single

There are two "Components" header in this section.

"a attribute" -> "an attribute".

  of attributes in the attribute-list shall be distinct. The keyword
  override shall not

The font of "override" is not correct.

  following the keyword override shall be the identifier for a attribute
  of the base datatype for the explicit

The font of "override" is not correct.

  A class-value denotes a value of a class datatype. When the class-value
  is a attribute-value-list,
  each attribute-identifier in the attribute-list of the class datatype
  to which the class-value
  belongs shall occur exactly once in the attribute-value-list, each
  attribute-identifier in the classvalue
  shall be one of the attribute-identifiers in the attribute-list of the
  class-type, and the
  corresponding independent-value shall designate a value of the
  corresponding attribute datatype. When
  the class-value is a value-list, the number of independent-values in

  the value-list shall be equal to
  the number of attributes in the attribute-list of the class datatype to
  which the value belongs, each
  independent-value shall be associated with the attribute in the
  corresponding position, and each
  independent-value shall designate a value of the attribute datatype of
  the associated attribute.

This whole paragraph is misleading and thus should be deleted. It assumes
the existence of a syntax for class-values. In the current version,
it is clear that there is no syntax for class-values.

  Operations: Equal, AttributeSelect.

"AttributeReplace" should be added to the list.

  Equal(x, y: class (attribute-list)): boolean If there exists an Equal
  method procedure for the class, then is
  Equal(x,y). Otherwise if there are no method procedures then is true if
  for every attribute-identifier f of the
  class datatype,

The term "method procedure" is not defined anywhere.

  attribute-type.Equal(AttributeSelect.f(x), AttributeSelect.f(y)), else
  false

The font of "false" is not correct.

  There is one AttributeSelect and one AttributeReplace operation for
  each attribute in the class datatype
  that is not a attribute procedure, of the forms:

"a attribute" -> "an attribute".

  There is one AttributeSelect and one AttributeReplace operation for
  each attribute in the class datatype
  that is a attribute procedure, of the forms:

"a attribute" -> "an attribute".

  notion of "subtype" or "subclass". A subtype of a Class datatype can =20
  have additional attributes (attributes); a subtype of a

```
We cannot understand why the word "attributes" is repeated in parentheses.
The second one "(attributes)" should be deleted.

  NOTE 3 An operation is represented by a attribute whose attribute-type

"a attribute" -> "an attribute".

<< 8.4.3 >>

  datatype (true, false), i.e., if s is a value of datatype set of (E),
  then s: E . B, and for any value e in the value

The mathematical definition of "s" is incomplete.  Probably, the dot
should be replaced by an arrow.

The set "B" (set of Boolean values) is used here without a definition.
We think such universal names should be defined somewhere in the document.

  Or(Not(IsIn(v,x)), IsIn(v,y)) = true, else false; i.e. true if and only
  if every member of x is a member of y;

This line should be indented.

  Setof(y: element-type): set of (element-type) is the function s such
  that s(y) = true and for all values v /= y,
  s(v) = false;
  i.e. the set consisting of the single value y;

There is an unnecessary line break.

  Select(x: set of (element-type)): element-type, where Not(Equal(x,
  Empty()), is some one value from the

A right parenthesis should be inserted at the end of the formula
"Not(Equal(x, Empty())".

<< 8.4.4 >>

  (E), then b: E . Z, and for any value e in the value space of E, b(e) =

The mathematical definition of "b" is incomplete.  Probably, the dot
should be replaced by an arrow.
```

The set "Z" (set of integer values) is used here without a definition.
We think such universal names should be defined somewhere in the document.

<< 8.4.5 >>

```
  Head(x: sequence of (element-type)): element-type, where
  Not(IsEmpty(x)), is the first value in the
  sequence x;
```

The font of "where" is not correct.

<< 8.4.6 >>

```
  Operations: Equal, Select, Replace.
```

The order of operation names is not consistent with the order of
the following operation definitions.

```
  Replace(x: array (index1, ..., indexn) of (element-type), y1: index1,
  ..., yn: indexn, z: element-type):
  array (index1, ..., indexn) of (element-type) is that value w of the
  array datatype such that w: (y1, ..., yn) -> z,
  and for all values p of the index product space except (y1, ..., yn),
  w: p -> x(p);
  i.e. Replace yields the function which associates z with the value (y1,
  ..., yn) and is otherwise identical to x.
```

There are unnecessary line breaks.

```
  type arrayB = array (1..m) of (array [1..n] of (integer));
```

The syntax of the inner array definition is not correct. Brackets
should be replaced by parentheses.

```
  Let A be a value of datatype array(array (index1, ..., indexn) of
  (element-type). For each index datatype
```

"array(array" should be changed to "array".

```
  Ord(x1: index1, ..., xn: indexn): ordinal is the ordinal value
  corresponding to the integer value:
```

The definition lacks the precise formula for the result value.

<< 8.6.1.2 >>

```
  value: the provision is associated the instantiation of a datatype8
```

The word "with" should be inserted after "associated".

```
  access: the provision is associated the access methods of a datatype
```

The word "with" should be inserted after "associated".

<< 8.6.1.3 >>

```
  scope-kind-value = "identifier" |
  "allidentifier" ;
  "recursiveidentifier" |
  "size" |
  "allsize" |
  "recursivesize" ;
```

The semicolon at the end of the second line should be changed to a vertical stroke.

```
  recursiveidentier: the provision is associated the all identifiers in
  all aggregate types, recursively
```

"recursiveidentier" -> "recursiveidentifier".

The word "with" should be inserted after "associated".

```
  recursivesize: the provision is associated the sizing parameters in
  all aggregate types, recursively
```

The word "with" should be inserted after "associated".

<< 8.6.1.4 >>

```
  subset-kind-value = "defined" |
  "undefined" |
  "*" |
  selecting-expr |
  value-expr ;
```

There is no syntax definition for "selecting-expr" nor for "value-expr".

<< 8.6.1.5 >>

```
  value-spec-value = "nil" |
  range-expr |
  selecting-expr |
  value-expr ;
```

There is no syntax definition for "range-expr", "selecting-expr",
nor "value-expr".

```
  -- selecting-expr: a selecting expression that limits the selection
  -- value-expr: a value expression that describes a pattern for the
  selection
```

The fonts of "selecting-expr" and "value-expr" are not consistent.

<< 8.6.1.6 >>

```
  -- range-expr: a range of values
  -- selecting-expr: a selecting expression that limits the range
  -- value-expr: a value expression that specifies the value
```

The fonts of "range-expr", "selecting-expr" and "value-expr" are not
consistent.

<< 8.6.4.1 >>

  Description: Specifies that the components of record or class type are
  ordered, unordered, or unspecified.

This description is misplaced. It is not an explanation of usage
triggers.

<< 9.1 >>

  to rename an existing datatype or name an existing datatype which has
  a complex syntax, or

A "to" should be inserted before "name".

```
  type-declaration = "type", type-identifier,
  [ "(" formal-type-parameter-list, ")" ],
  "=", [ "new" ], type-definition |
  normative-datatype-definition ;
```

The indentation of the last line should be corrected.

"normative-datatype-definition" is not a correct non-terminal name.
It should be "normative-datatype-declaration" according to 9.4.

<< 9.1.2 >>

  The type-definition defines the value space of the new datatype
  (family) --- there is a one-to-one
  correspondence between values of the new datatype and values of the
  datatype described by the typedefinition.

In many cases, "one-to-one" is written as "1-to-1". Please be
consistent.

  characterizing operations is possible. For example, acceleration and
  velocity may have identical computational value
  spaces and operations (datatype real) but quite different physical ones.

We do not understand what is meant by "ones" at the end of the
sentence.

<< 9.3 >>

```
  parameter-type = type-specifier ;
  parameter-name = identifier ;
```

Syntax rules should be aligned properly at the position of equality
symbols. The position of equality symbol in these rules are not the
same as that of "parameter".

<< 9.5.1 >>

```
  URI-or-type-identifier = URI |
  identifier ;
```

There is no syntax definition for "URI".

```
  tag-type = type-specifier ;
  discriminant = value-expression ;
```

These two syntax rules are not referred to. In 8.3.1, there are
references to "tag-type" and "discriminant", but 8.3.1 has its own
definitions for these non-terminals.

```
  Components: The source value identifies a resource that contains a
  program-text. Each declaration in that
```

The term "sources value" appears without any explanation.

<< 9.5.2 >>

```
  macro-definition = "macro", identifier, "(" param-list ")" ,
  "{", text, "}";
```

There is no syntax rule for "param-list".

<< 10.1.5, Note 4 >>

```
  type editcharacter = character({iso standard 646}) selecting
  ('0'..'9', '.', ',', '+, '-', '$', '#', '*'),
```

The type "character" is unordered. Is it possible to write
a range notation "0".."9" for this type?

<< 10.1.9 >>

```
  type private(length: NaturalNumber) = new array (1..length) of (bit)
```

"NaturalNumber" -> "naturalnumber".

<< 10.1.10 >>

```
  provisions of ISO 7350:1991 whose registration-number is the value of
  registry-index. The form of the
```

"ISO 7350" should be changed to "ISO/IEC 7350" according to Annex A.

```
  provisions of ISO 10036:1991 whose registration-number is the value of
  registry-index. The form of the
```

"ISO 10036" should be changed to "ISO/IEC 10036" according to Annex A.

```
  NOTE 2 ObjectIdentifier is treated as a primitive type by many
  applications, but the mechanism of definition of its value
```

"ObjectIdentifier" -> "Objectidentifier".

<< 10.2.1 >>

```
  Components: element shall any datatype.
```

A verb should be inserted after "shall".

<< 10.2.3 >>

```
  if And(IsPresent(x), IsPresent(y)), then Binary-op(Cast.base(x),
  Cast.base(y)),
  else undefined.
```

There is an unnecessary line break in this definition.

<< 11.1, Note 2 >>

```
  datatype generator to the mapped parametric datatypes. In this way,
  property (i) above may be satisfied for internal
  generated datatypes.
```

What is meant by "(i)"?

<< 12 >>

```
  12 Annex A (informative): Character-set standards
```

In this document, annexes have two section numbers. We think that
only the alphabetical form "Annex A, B, ..." suffices.

```
  codes defined by the repertoire is outside of the scope of this
  International Standard .
```

"outside of" -> "outside".

```
  ISO/IEC 4873:1991 Information technology --- ISO 8-bit code for
  information interchange ---
```

  Structure and rules for implementation

There is an unnecessary line break in the name of the standard.
The same error is found in many standard references in the succeeding
text.

  ISO 6861: --- Information and documentation --- Cyrillic alphabet coded
  character sets for historic
  Slavonic languages and European non-Slavonic languages written in a
  Cyrillic script, for bibliographic
  information interchange

The number of the standard is incomplete. The published year should
be given after the colon. The same error is found in many standard
references in the succeeding text.

  The following are International Standards for character-set
  registration. Character sets registered under the

This line should not be indented.

<< 13 >>

  13 Annex B: (informative) Recommendation placement of annotations

The title seems to be grammatically incorrect.

<< 14.3 >>

  v = S . M . RE

The formula is not correct. "R" should be raised to the power "E".

  M is the mantissa, either zero or a value of the datatype scaled(radix,
  precision) range(radix ^ -
  precision, 1) excluding(1).

The operator "^" is undefined.

  denorm, with the requirement that denorm = false implies d = R-1 and
  denorm = true implies d = Rprecision.

The symbol "d" is defined here, but where is it used?

The power "precision" should not be placed in a separate line.

v = S x M x R-P

The formula is not correct". "R" should be raised to the power "-E".

Why do you use a symbol for multiplication different from that in
14.3?

<< 14.5 >>

Tag is a type-attribute which specifies whether and how the tag-value
of a value of a value of a choice

One of two "a value of"s should be deleted.

<< 15.1.9 >>

has the value -maxint.r(-f) and maxrf has the value maxint.r(-f). A
scaled datatype with the corresponding

The formula is not correct. "r" should be raised to the power "(-f)"
(two places).

minimum and maximum values (and any sub-type thereof) is mapped to the
Pascaltype integer, with each

"Pascaltype" -> "Pascal type".

scaled value N.r(-f) being mapped into the Pascal integer value N. In
order for the characterizing operations

The formula is not correct. "r" should be raised to the power "(-f)".

<< 15.1.10 >>

The LI datatypes real range(rmin..rmax) and real(radix, precision)
range(rmin..rmax) map to the Pascal type

"LI datatypes" is an old terminology.

<< 15.1.11 >>

```
  procedure Squareroot(x: complex; var t: complex);
```

The name of the function should be "SquareRoot" ("R" capitalized)
which is the name defined in 8.1.11.

<< 15.2.7 >>

```
  function IsEmpty(var s: sequenceoftype): Boolean;
  begin IsEmpty := eof(s) end;
```

"reset(s);" should be inserted as the first statement of this
function. If you insist that the file is always reset after an
operation, it is not consistent with the following definition:

```
  procedure Head(var s: sequenceoftype; var t: mapped-type);
  begin reset(s); read(s, t); reset(s); end;
```

"reset(s);" as the first statement of this function would be unnecessary.

```
  continue := mapped-typeEqual(s^, t^);
```

"mapped-typeEqual" -> "mapped-type.Equal".

  Because a Pascal file-type, however, cannot be the component-type of
  another file-type, LI datatypes of the

"LI datatypes" is an old terminology.

<< 15.3.4 >>

  sequence datatype implementation in D.2.7, and certain size-subtypes
  are mapped to specific Pascal types in
  E.4.

The section reference "E.4" is not correct. "E" is the section for
MUMPS, and Pascal types are not given there.

<< 15.4.2 >>

  The LI datatype modulo(modulus) maps to the Pascal subrange type
  0..modulus-1, according to the mapping

"LI datatype" is an old terminology.

<< 15.4.4 >>

```
  type bitstringsizek = packed array [1..k] of Boolean;
```

In the printed text, there is no space between "type" and
"bitstringsizek".

<< 15.4.5 >>

```
  type charstringsizek = packed array [1..k] of char;
```

In the printed text, there is no space between "type" and
"charstringsizek".

<< 15.4.6 >>

  datatypes (see E.1.9). The scalarMultiply operation is mapped to

The section reference "E.1.9" is not correct. "E" is the section for
MUMPS, and Pascal types are not given there.

<< 15.4.8 >>

```
  type octetstringsizek = packed array [1..k] of octet;
```

In the printed text, there is no space between "type" and
"octetstringsizek".

<< 15.5.4 >>

  An GPD datatype of the form optional(T) can only be mapped to Pascal if
  the type T can be mapped to

"An GPD" -> "A GPD".

  NOTE Alternatively, optional(T) can be mapped to ^mappedT, where
  mappedT is the mapping of LI datatype T into

"LI datatype" is an old terminology.

<< 15.6.2 >>

An GPD datatype declaration which declares a single datatype (no
  parameters) can be mapped to Pascal as

"An GPD" -> "A GPD".

  An GPD datatype declaration which declares a family of datatypes, using
  one or more parameters, cannot, in

"An GPD" -> "A GPD".

<< 15.6.3 >>

  An GPD generator declaration cannot, in general, be mapped into Pascal.
  In many cases, however, each

"An GPD" -> "A GPD".

<< 16.1.6 >>

  NOTE An alternative is to map date and time values to a character
  string in $H[OROLOG] format, which has the form

What is meant by "$H[OROLOG]"?

<< 17.1 >>

  the interface specification. For example, Sequence is a native datatype
  in GPDSP, and Set is a native

The term "GPDSP" is not defined.

<< 17.4 >>

  Tree, or the GPDSP-characteristic indefinite-list datatype.

The term "GPDSP" is not defined.

<< 17.5.5 >>

  datatypes as a choice datatype one of whose alternatives is the true
  datatype of the column and and the other

One of two consecutive "and"s should be deleted.

  modelled as having choice datatypes. "Void" was originally called
  "Null", but has been renamed to avoid

The word "modelled" should be spelled as "modeled" to be consistent
with other uses of this word in this document.

  There is consensus that Undefined is not a datatype. Undefined is a
  part of the behaviour of entities which

The word "behaviour" should be spelled as "behavior" to be consistent
with other uses of this word in the document.

<< 17.5.7 >>

  applications. "Pure" pointer datatypes can be modelled as: pointer to
  (T) excluding (null).

The word "modelled" should be spelled as "modeled" to be consistent
with other uses of this word in this document.

<< 17.5.11 >>

  Array-types whose values have different numbers of elements (Ada
  [1:?n]). Such types are designated

Ada syntax does not allow the use of a question-mark.

<< 17.7.1 >>

  Issue 35. Should NaturalNumber or Unsigned be GPD datatypes?

It would be better to change "NaturalNumber" to "Naturalnumber".

<< 17.7.4 >>

  Issue 2. Should Character-string types be ordered?

This issue number is not correct. It is not ascending, and the
same number has already been used.

<< 17.7.5 >>

```
  mathematical operations. Thus Tensor is outside the scope of the LI
  datatypes.

"LI datatypes" is an old terminology.

  state, position, etc., attributes, goes beyond the scope of this
  standard. The datatype, its attributes and

The position of "attributes" looks strange.

<< 17.8 >>

  Issue 41. How much of the concept "mapping onto the LI datatypes"
  should be standardized?

"LI datatypes" is an old terminology.

  values of all "parameters" of the LI datatypes, and a discussion of the
  distinction between "logical

"LI datatypes" is an old terminology.

  constructions which equate to various LI datatypes might be quite
  complicated.

"LI datatypes" is an old terminology.

<< typographical error throughout the draft >>

A hyphen is often inserted in a word which should normally have
no hyphens in it.  These hyphens should be deleted.

8.2, "sub-type"
  relationship between the value spaces of the base datatype and the
  sub-type.

8.2, "sub-type"
  informal name for the sub-type generator, and the subtype generator is

8.3.3, Note 5, "out-side"
  distinctions relate to the methods of moving values between program
  elements, which are out-side the scope of this
```

```
8.4.1, Note 3, "sub-type"
  record datatype) is not a sub-type of the base record datatype: none of

8.5, "there-by"
  datatype or datatype generator there-by defined. The
  actual-type-parameters, if any, shall correspond in

12, "in-terpreted"
  purposes. Whether "character(repertoire)" is in-terpreted as requiring
  the characters to be represented by the

12, "order-ings"
  values used in a particular implementation of the language. Such
  order-ings have no semantics with respect

13.3, "procedure-at-tributes"
  procedure-at-tributes should be distinguishable from type- or
  component- attributes by their text.

15.1.9, "sub-type"
  minimum and maximum values (and any sub-type thereof) is mapped to the
  Pascaltype integer, with each

15.2.1, "other-wise"
  and is not other-wise required. Each select-item in the select-list
  which is a single value is mapped to the

15.2.3, "Pas-cal"
  Terminations other than normal are not supported by Pascal, and no
  procedure datatype involving them can
  be mapped into Pas-cal.

15.4.4, "effi-cient"
  although more effi-cient structures for bitstring can be developed.

15.5.2, "in-tended"
  implementation choices, depending on the in-tended searching
  strategies, i.e. the true "characterizing operations" of the type.

16, "re-verse-inward-mapping"
  otherwise stated the re-verse-inward-mapping is the inverse of the
  inward-mapping, using the necessary
```

```
16, "con-versions"
  operations, requires the programmer to perform the appropriate
  con-versions. The GPD datatypes involved

17.1, "pro-vides"
  procedure calling, pro-vides the procedure call model, the requirements
  for interface specifications and the

17.2, "map-ping"
  Such a map-ping violates the notion of semantic equivalence of the
  datatypes.

17.3, "mathe-matical"
  consensus that mathe-matical datatypes should be defined by appeal to
  standard mathematical references.

17.5.3, "pro-grams"
  these semantics, is the most frequently occurring datatype in COBOL
  pro-grams, and also appears in other

17.5.6, "ma-chines"
  "Excluding" subtypes, the same GPD datatype as implemented by two
  ma-chines might actually have nonisomorphic

17.5.7, "sup-ported"
  which the handle refers is intentionally not sup-ported, while
  accessing the object to which a pointer refers is a

17.5.8, "There-fore"
  position. There-fore, the ordering of fields in a Record is not a
  property of the conceptual datatype itself.

17.5.11, "sub-scripting"
  passed is either a caller-defined sub-scripting function or a set of
  parameters by which the called subprogram

17.7.4, "de-fine"
  Some programming languages make the character-string primitive in order
  to de-fine useful operations that

<< typographical error throughout the draft >>
```

A space just after a hyphen should be deleted in the following places.

6.8.6, "single- valued"
value. Such a mapping is required to be single- valued, i.e. there is

8.1.7, "digit- string"
the digit- string interpreted as a decimal number. If the negative-sign

11, Note 1, "object- types"
manipulate specific object- types, which have specific datatypes
expressed in a specific language or languages. The

13.3, "component- attributes"
procedure-at-tributes should be distinguishable from type- or
component- attributes by their text.

15.1.3, "enumerated- value-list"
type (enumerated- value-list). Each enumerated-value is mapped to the
Pascal value with the corresponding

15.1.11, "implementation- defined"
implementation- defined, and then only if rmax and the given or default
radix and precision parameters define

15.2.1, "alternative- type"
the alternative- type maps to a Pascal record-type, then the
corresponding mapped-type is: (all-fields-of-the-

15.2.1, "record- type"
alternative-type does not map to a Pascal record- type then the
corresponding mapped-type is: (mapped-fieldidentifier

15.2.7, Note, "mapped- type"
some Pascal type mapped- type, as specified in this Annex, can also be
mapped into Pascal using the type:

15.2.8, "array- type"
corresponding Pascal array- type by mapping the value of each element
of the array-value to its

17.3.1, "well- defined"
datatype is well- defined. Specifically, the Pascal ORD operation on
enumerated types is not characterizing -

```
17.5.5, "null- valued"
  distinct from those of any other primitive type. The SQL2 null- valued
  column is properly described in GPD

17.5.11, "language- independent"
  can reconstruct the subscripting function. In a language- independent
  interface, in order for the two language

17.5.11, "dependent- values"
  be made explicit. Thus, this case is a special case of "conformant"
  arrays using "dependent- values" which are

<< typographical error throughout the draft >>

The formatting of the header "Syntax:" is often inappropriate.
For example, in 7.5.2, it should be separated from the preceding
text.  In 7.6, it becomes a part of the preceding line, and should
of course be separated.  Many sections do not have this header,
for example in 8, 8.1, ...  We like to have a consistent usage of
the header.

<< typographical error throughout the draft >>

The character "s" at the end of a noun, meaning plural, should be
in the normal font, even when the noun itself is a technical term
and is typeset in some special font.

8.1, "parameter-names"
  The operation-name is an identifier unique only within the datatype
  being defined. The parameter-names

9.1, "actual-type-parameters"
  actual-type-parameters which must appear in a type-reference which
  references this type-identifier.

9.1.2, "formal-type-parameters"
  parameter-list is present, then the type-identifier is declared to
  identify a family of datatypes
  parameterized by the formal-type-parameters.

9.3, "parameter-names"
  The parameter-names of the parameters in a termination-parameter-list
  shall be distinct. No

10.1.4, "bit-literals"
  there are no bit-literals in the bitstring-literal, then the value
  denoted is the sequence of length
  zero.

<< typographical error throughout the draft >>

In syntax rules, syntactic items should be separated by a comma,
but this comma is often missing.

8.1.4, after ","
  repertoire-list = repertoire-identifier,
  { "," repertoire-identifier } ;

8.4.4, after "independent-value"
  value-list = "(", independent-value
  { ",", independent-value }, ")" ;

8.6, after "actual-parameter"
  actual-parameter-list = actual-parameter { ",", actual-parameter } ;

9.2, after "value" and after "="
  value-declaration = "value" value-identifier, ":", type-specifier,
  "=" independent-value ;

9.5.1, after "including", after "(", after "select-list"
  import-type = "import", URI-or-type-identifier,
  { "including" "(" select-list ")" |

9.5.1, after "excluding", after "(", after "select-list"
  "excluding" "(" select-list ")" } ;

9.5.2, after "(", after "param-list"
  macro-definition = "macro", identifier, "(" param-list ")" ,
  "{", text, "}";

10.1.5, after " "
  character-name = identifier, { " " identifier } ;

10.1.10, after "identifier", after "(", after "numberform"
  nameandnumberform = identifier "(" numberform ")" ;
```

```
10.1.10, after "registry-name"
  collection-identifier = registry-name registry-index ;
```

Netherlands

National Committee	Clause/ Subclause	Paragraph Figure/ Table	Type of comment (General/ Technical/Editorial)	COMMENTS	Proposed change	OBSERVATIONS OF THE SECRETARIAT on each comment submitted
NL1			general	Sometimes multiple notes in a (sub)section are numbered (as in 6.4) sometimes not (as in 6.2).	Unify throughout the document.	
NL2	Introduction	First bullet	ed	the text seems to exclude the use of 11404 in programming language context	Suggestion: "... this I.S. is also used for formal ..." and "(see for instance ISO/IEC 11179-3)"	
NL3	Introduction	Fourth bullet	ed	Usage of "thus" (twice)	Replace "thus" by "so that" (or "thus it is possible" by "thereby making it possible")	
NL4	Introduction	Fifth bullet	ed	The usage of "(1)", "#1" and then (in the last sentence) "(1)" again is confusing		
NL5	Clause 3		ed	To have two definitions for parametric value in 3.41 and 3.42 is strange	Merge the definitions into one definition with a clear indication when either of them is to be used.	
NL6	Clause 3.45		tech	primitive internal datatype is defined in relationship with programming languages, and used in the mapping section; are mappings limited to programming languages? If not, the definition should be changed		
NL7	Clause 3.50		ed	Awkward use of the notion "sign"; why not use the original "image"?		
NL8	Clause 3.53		ed	It might be useful to consider to add a note similar to the last note of 6.2 to the definition of sentinel value in 3.53.		
NL9	Clause 3.60		ed	Error in defin ition.	"bounded below" must be replaced by "bounded above".	
NL10	Clause 4.4		tech/ed	This section uses "GDP program text conformance" in the header, "program" in the 1st sentence and "program text" in the Note. Is this about "Conformance of a GDP program text", or "Text conformance of a GDP program"? Should the 1st sentence be "A GDP program text that conforms ..."?		

National Committee	Clause/ Subclause	Paragraph Figure/ Table	Type of comment (General/ Technical/Editorial)	COMMENTS	Proposed change	OBSERVATIONS OF THE SECRETARIAT on each comment submitted
NL11	Clause 4.4		tech	This clause requires that a conforming GDP program text implements ALL of the requirements in clauses 5-10, whereas direct conformance (4.1) 'only' requires a subset to be conforming (no conformance requirements for unused features).	Some more explanation, in the form of a Note is required	
NL12	Clause 5.2		ed	Mixing the various fonts gives sometimes a messy look	Consider to increase the fontsize of the special fonts	
NL13	Clause 6.1		tech/ed	LID had "In this I.S., characterizing operations are purely informative and have no normative impact."; this sentence and the subsequent note are removed. There used to be much confusion about the normativeness of the characterizing operations, and hence the original sentence.	Suggest to reinsert the old text	
NL14	Clause 6.1	para 2	ed	Last sentence is grammatically incorrect.	Reformulate: "refers to datatypes" or "is used to mean datatypes"	
NL15	Clause 6.2	2nd para after 1st note	ed	Readability.	Replace (twice) "(elements of a value space" by "(those elements of the value space ..."	
NL16	Clause 6.2	last note	ed	Readability.	Reformulate last sentence: "For those sentinel values the mentioned characterizing operations are not defined."	
NL17	Clause 6.3		tech	The mentioned properties only apply to regular values from the value space; this should be mentioned (example: adding the sentinel value NaN to the real values does not make the real value space unordered).		
NL18	Clause 6.4	Note 1	ed	Seems to be a left over of earlier text; it has now no relation with its context.	Either add context or remove the note.	
NL19	Clause 6.6	Note 1	tech	Why is "designations" better that "symbols"? If it is better, should "designations" not be added to the terminology section?		
NL20	Clause 6.8.4		tech	Where is this used? Only in 8.6.3? The name "identifier uniqueness" is dangerous: 8.1 has "The operation-name is an identifier unique only within the datatype being defined"; this is probably not what is being specified in 6.8.4.		
NL21	Clause 6.8.9		ed	Wording	Replace "to have values in a valid value" by "to have a valid value"	
NL22	Clause 6.9	1st and 2nd para	ed	Old text?	Remove	

National Committee	**Clause/ Subclause**	**Paragraph Figure/ Table**	**Type of comment** (General/ Technical/Editorial)	**COMMENTS**	**Proposed change**	**OBSERVATIONS OF THE SECRETARIAT** on each comment submitted
NL23	Clause 7.3.1		tech	Explain the notion "ISO/IEC-10176-extended-letter"		
NL24	Clause 8.1.2, 8.1.3		tech	The Operations section lists the Equal operation, defined on part of the value space: not on the value-space-source part of the value space. Does this make these latter values sentinel values? If so, should this be mentioned? Also applies to clause 8.1.3.		
NL25	Clause 8.1.2		ed	Example 2 belongs in clause 8.1.3	Move example.	
NL26	Clause 8.1.3		ed	In the operations section: enum-value-list is the wrong non-terminal name (5 times).	Replace "enumerated(enum-value-list)" by "enumerated(enumerated-value-list)" (5 times)	
NL27	Clause 8.1.10	Parametric values	tech	The value of "factor" must be greater than 0.	Reword to read: "… and factor shall have an integer value greater than 0".	
NL28	Clause 8.1.10		ed?	Wrong symbol for R		
NL29	Clause 8.1.11	Parametric values	tech	The value of "factor" must be greater than 0.	Reword to read: "… and factor shall have an integer value greater than 0".	
NL30	Clause 8.2.6		tech	"plus" and "sentinel" should go together.	Change the syntax for the extended-type to: extended-type = base, ["plus", "sentinel"]	
NL31	Clause 8.4.1 (and others)		ed	Forward reference.	add forward reference for the provision-statement	
NL32	Clause 8.4.2		ed	Layout muddled.	Alignment of sytax rules, empty Description section, 2 Components sections	
NL33	Clause 8.4.2		ed	Check the text in the Operations section.	Indentation and use of empty lines can be improved; spelling should be checked ("of the forms" in stead of "of the form")	
NL34	Clause 8.4.2		ed	Text Operations section, equal operator.	Reformulate "... no method procedures then is true if for ...".	
NL35	Clause 8.4.3		ed	Wrong symbol	values: the "->" symbol	
NL36	Clause 9.4		ed	Explanation needed on the use of the concept normative-datatype-definition.		
NL37	Clause 9.5.1			What is "the source value"?		
NL38	Clause 9.5.2		tech	There is no usage or mentioning in the document of the macro concept		

United Kingdom

GB	P 70		ed	The line 'Let A be a value of datatype array(array (index1, ..., indexn) of (element-type)' has an unbalanced left parenthesis.		
GB	passim		ed	There are a few word processor-induced errors. Page 133 has 'pro-gram' (reminiscent of Chick's Own, for those old enough to remember it). There are things like 'single-valued' (p 18), 'digit- string' (p 38), 'null-valued' (p 133). It would be easier to search for them from scratch than for me to give a full list.		
GB	p 71:		ed	Fortran standard reference should be ISO/IEC 1539-1:2004 (but see below).		
GB	p 70-71:	Note 6	ed	is confusing rather than enlightening. It semi-formally defines a sample mapping of array indices to value location (as one of 'many such' mappings), gives a bit of unnatural Fortran code and then weakly says that 'The Fortran standard ..., however, requires a mapping function which gives a different sequence representation from that given in Note 6'. A reader who does not know about Fortran array indexing will be left wondering what the point of the example was; so will a reader who does know about it, since the point could be made more succinctly in far fewer words. However, if the point is to be made at length, I would offer these edits.	Note 6 lines 2-3: replace 'There are many such functions' by 'There are various possible such functions, some of which are explicitly defined in the standards for particular languages". line 14: replace 'The Fortran declaration:' by 'In Fortran the standard specifies that multidimensional arrays are stored with the left-most varying most rapidly. Thus in the following declaration:' line 16: before 'declares' insert 'which' line 17+: Insert new text 'In order that the elements of a single row be contiguous for possible use in a language which uses the mapping above, it is necessary to write the Fortran code in a counter-intuitive manner.'. line 18: Replace 'And the' by 'The' lines 26-27: Delete: 'The Fortran standard ... Note 6." If the reference to the Fortran standard is required, add it at the edit above.	
GB	p 133:			'also appears in other standard languages, such as PL/I' This is the only mention of PL/I (latest standard dated 1979) in the document. Is it a good exemplar?		

GB	clause 5.1	(page number 10 of the document, PDF page 22) heading of Table 1		The document includes ISO/IEC 10646-1:2000 in its Normative References, and here refers to ISO/IEC 10646, and possibly elsewhere. The most recent edition of ISO/IEC 10646 was published in 2003 as a single-part standard.	The references should be checked for continued applicability and updated as appropriate.	
GB	Clause 6.5.1			This states that the concept of equality is defined for all datatypes. This may be true within the the scope of this standard, but in ISO/IEC 9075, Database language SQL, there are some datatypes, specifically user-defined types, for which equality is not defined. The same may be true in some object-oriented languages. This situation may merit the inclusion of an informative Note.		
GB	clause 6.3			The definition of Bound in 3 defines the bound to be a member of the datatype. Whilst this is likely to be the case for types with finite cardinality, it may not necessarily be true for types with non-finite cardinality. Consider the datatype consisting the reciprocals of positive integers. The set is bounded below by zero, which is not a member of the set.		
GB	clause 6.4	unnumbered Note		Is this correct? Consider the rational numbers. They can be mapped onto the integers in at least one well-known way, and given any n, I can find mappings such that each integer of length not greater than n digits is associated with a rational number, and still have rational numbers left over.		
GB	Passim			The document contains informative references to several standard programming languages (including FORTRAN, C, Ada, Pascal) and to other standards (e.g.PHIGS). It would be useful if the full references to these standards could be collected into an informative annex, similar in style to the Normative References clause.		
GB		The Table of Contents		There are dual numbering for clauses in the Annexes. I assume that this is an artifact of the process by which the current draft was created and there will be only single numbering in the FDIS.		