

ISO / IEC JTC1 / SC22

Programming languages, their environments and system software interfaces  
Secretariat: CANADA (SCC)

**ISO/IEC JTC1/SC22  
N 1354**

**MAY 1993**

**TITLE:** Summary of Voting and Comments received on  
Second CD 11404: Language-independent  
Datatypes

**SOURCE:** Secretariat ISO/IEC JTC1/SC22

**WORK ITEM:** JTC1.22.17

**STATUS:** New

**CROSS REFERENCE:** N1305

**DOCUMENT TYPE:** Summary of CD ballot

**ACTION:** For information to SC22 Member Bodies.  
See attached.



SUMMARY OF VOTING ON:

Letter Ballot Reference No: SC22 N1305  
Circulated by :JTC1/SC22  
Circulation Date :1993-01-07  
Closing Date :1993-04-16

SUBJECT: Second CD11404: Language-independent Datatypes

---

**The following responses have been received:**

'P' Members supporting CD registration, without comments	: 08
'P' Members supporting CD registration, with comments	: 02
'P' Members not supporting the CD registration	: 02
'P' Members abstaining	: 02
'P' Members not voting	: 07

---

**Secretariat Action:**

The Secretariat will forward the attached comments to WG11 for consideration and recommendation on further processing of CD11404.

ISO/IEC JTC1/SC22 LETTER BALLOT SUMMARY

PROJECT NO: JTC1.22.17

SUBJECT: Second CD11404: Language-independent Datatypes

Reference Document No: N1305  
Circulation Date: 1993-01-07

Ballot Document No: N1305  
Closing Date: 1993-04-16

Circulated To: SC22 P, L

Circulated By: Secretariat

SUMMARY OF VOTING AND COMMENTS RECEIVED

	Approve	Disapprove	Abstain	Comments	Not Voting
<b>'P' Members</b>					
Austria	( )	( )	( )	( )	(✓) <i>not counted</i>
Belgium	( )	( )	( )	( )	(✓)
Brazil	(x)	( )	( )	( )	( )
Bulgaria	( )	( )	( )	( )	(✓)
Canada	( )	( )	(x)	( )	( )
China	(x)	( )	( )	( )	( )
Czech/Slovak Re	( )	( )	( )	( )	(✓)
Denmark	(x)	( )	( )	( )	( )
Finland	(x)	( )	( )	( )	( )
France	(x)	( )	( )	( )	( )
Germany	( )	( )	(x)	( )	( )
Greece	( )	( )	( )	( )	(✓)
Italy	(x)	( )	( )	( )	( )
Japan	( )	( )	( )	( )	(✓)
Netherlands	(x)	( )	( )	( )	( )
New Zealand	(x)	( )	( )	(x)	( )
Romania	(x)	( )	( )	( )	( )
Slovenia	( )	( )	( )	( )	(✓) <i>not counted</i>
Sweden	( )	(x)	( )	(x)	( )
Switzerland	( )	( )	( )	( )	(✓)
UK	(x)	( )	( )	(x)	( )
USA	( )	(x)	( )	(x)	( )
USSR	( )	( )	( )	( )	(✓)
<b>'O' Members</b>					
Argentina	( )	( )	( )	( )	( )
Australia	( )	( )	( )	( )	( )
Cuba	( )	( )	( )	( )	( )
Hungary	( )	( )	( )	( )	( )
Iceland	( )	( )	( )	( )	( )
India	( )	( )	( )	( )	( )
Poland	( )	( )	( )	( )	( )
Portugal	( )	( )	( )	( )	( )
Singapore	( )	( )	( )	( )	( )
Turkey	( )	( )	( )	( )	( )
Thailand	( )	( )	( )	( )	( )
Yugoslavia	( )	( )	( )	( )	( )



UK COMMENTS ON SC 22 N1305 : LANGUAGE INDEPENDENT DATATYPES

The UK votes YES with the following comments:

1. The definition of datatype in section 3.11 and 6.1 is a) inconsistent and b) does not make clear that characterising operations are non-normative:

The definition of datatype in section 3.11 and 6.1 should be changed to "a collection of distinct values and properties of those values. It is characterised by a collection of operations on those values. The characterising operations are only normative within the context of mappings within this draft International Standard."

2. There is an implication that most entities will comply directly with this standard, but it is unclear how they can do so:

An additional note should be added to 5.2: "Existing standards for programming languages are expected to provide for Indirect Compliance rather than Direct Compliance."

3. Clarification is needed that the codes associated with characters in 8.1.4 are not being used:

In 8.1.4 add "whose code are" after "character-set" in the Example.

4. It is suggested that an outward mapping be produced, preferably in another language in addition to that of Pascal, in order to demonstrate how this differs from an inward mapping.
5. It is recommended that an ad-hoc meeting be set up at the SC22 Plenary for WG11 to discuss with other WGs issues arising from this and its other cross-language standards.

**ITS** Informationstekniska  
standardiseringen

Klaus Appel

1993-04-14

ITS 21/22

Swedish comments to CD 11404.2

1. Clause 2:

After "ISO 8601:1988" add "including Technical Corrigendum 1:1991".  
For your information, an Amendment 1 is presently at DAM stage.

2. All character names should follow JTC1/SC2 standards, particularly ISO/IEC 10646-1 (presently being printed). I refer to Annex J Rule 1: convention, only Latin capital letters A to Z, space, and hyphen shall be used for writing the names of characters."

Clause 4.1 under terminal symbol "quotation-mark" -> "QUOTATION MARK"  
"apostrophe-mark" -> "APOSTROPHE"

under non-terminal symbol "hyphen mark" -> "HYPHEN-MINUS"

under repeated sequence "open-brace mark" -> "LEFT CURLY BRACKET"  
"close-brace mark" -> "RIGHT CURLY BRACKET"

under optional sequence "open-bracket mark" -> "LEFT SQUARE BRACKET"  
"close-bracket mark" -> "RIGHT SQUARE BRACKET"

under alternative sequence "vertical stroke mark" -> "VERTICAL LINE"  
"full-stop mark" -> "FULL STOP"

under production "equal-sign" -> "EQUALS SIGN".

Clause 7.1 under iii): "ISO 10646:1992" -> "ISO/IEC 10646-1:1993"

Clause 8.1.4 Note 4: "Latin alphabet" -> "Latin alphabet No. 1"

Clause 8.1.6 under value-syntax: "a single decimal point" -> 'the letter "T" and/or a single decimal point'. Explanation: according to ISO 8601 clause 5.4.1, a time-designator "T" shall be used between date and time in a combined date and time representation.

## U.S. comments on LI Datatypes (ISO CD 11404.2)

The U.S. votes NO on the progression of CD 11404 to DIS. Our vote will be changed to YES if the two comments designated Major below are satisfactorily resolved.

Issue: 1

Subject: Define the useful model of Array-types

Type: Major

Clauses: 8.4.4 and 8.4.5

Xref: RPC 9.5.2.3 and X3T5.5/93-020

Rationale: The Sequence and Array types of LID cover at least the following conceptually distinct datatypes:

Fixed-length single index - vector. What must be preserved is the ordering of the components, indexing could be off by 1 in language mappings without serious consequences.

Variable-length single index - sequence (values of different lengths). What must be preserved is the ordering of the components, indexing is effectively ordinal, so any language convention is adequate.

Fixed multiple indices - matrix/map. What must be preserved is dimensionality (shape) and ordering within each dimension. Preservation of indices is desirable, but not required. Total linear ordering is not conceptually meaningful.

Conformant multiple indices -matrix/map template. This is a template, not a datatype, and is satisfied by any datatype which can "conform" to the template syntax. Alternatively this could be represented as a sequence of sequences in which values must conform to some unstated rules. The conformant array syntax is used to facilitate mapping to programming languages.

In addition, the following two "datatypes" exist in Fortran:

Multiple indices of which the last is variable. This appears to be properly modelled as a sequence of (array of n-1 dimensions).

Reshaped arrays, in which the size is constant, but the indexing and dimensionality is different between the caller and the called procedure. This appears to be properly modelled as a sequence, since the ordering of ALL the elements must be shared, while indexing is a local matter.

Finally the notion "slicing" has been suggested as an important concept for LIPC/RPC. Conceptually the interface argument type of a "slice" is that of the result of the slicing operation, which is an "ordinary" array-type, and the means of generation of the "sliced value" is a caller-only (marshalling) issue. In practice, where the slice is implemented by a subscripting procedure ("thunk") provided by the caller and invoked by the called procedure, that should be declared to the called procedure by an annotation.

What is clear from the above is that the notion that an array is a Map from one or more index-types to the element-types is a poor characterization of the semantics of the array-type when it is mapped into multiple programming languages. The index-value to element-value association does NOT have to be preserved. It is at best desirable in ONE of the above cases. That the access is indexed (6.8.5) is an important property, and the dimensionality and the ordering of values WITHIN a dimension must be preserved.

Moreover, the only distinction between sequence and array as parameter types seems to be that the size of an array dimension must be formally specified (hence all of the dependent-value stuff) while the size of a sequence value is somehow implicitly carried in the value. Noting that only values can actually be passed, this distinction doesn't seem to be useful. Thus LIPC does not

need both Sequence and Array.

Proposed Change:

- a. Merge Sequence and Array.
- b. Explain the meaning of Array in terms of the CARDINALITY of the index-types, and the ORDINAL mapping from index-values to array elements.
- c. Devise a notation for variable-length dimensions and a standard mechanism for handling length parameters that are effectively supplied at runtime. Size parameters are part of the value of an array-type which has one or more variable dimensions, just as they are of a variable length sequence.

Issue: 2

Subject: Make Table model support SQL model

Type: Major

Clauses: 8.4.6

Xref: SC21/WG3 N1452 and X3T2/92-111

Rationale: The existing table-type is too restrictive to support the SQL table model and can be generalized to support the SQL model as well as the notion currently supported by LID.

Proposed Change: Redescribe Table as follows:

Description: Table generates a datatype, called a table datatype, whose values are associations among values in the product space of one or more field datatypes.

Syntax: table-type = table (“ field-list “) .

Components: (same as Record).

Values: the value space of Table(*field-list*) is identical to the value space of Bag(Record(*field-list*)).

Value-syntax:

table-value = empty-value | (“ table-entry { “,” table-entry } “) .

table-entry = record-value .

Properties: non-numeric, unordered, exact if and only if all field-types are exact, heterogeneous, variable size, no uniqueness, no ordering, access is keyed, dimensionality is two.

Subtypes:

- i) any table-type which has the same number of fields in the field-list, the field-names correspond, and each field-type is the same as, or a subtype of, the field-types of the base datatype, or
- ii) (as is)

Operations: Equal, Empty, Insert, Delete from Bag; Select

Select(*x*: table(*field-list*), *p*: procedure(in row: record(*field-list*)): boolean): table(*field-list*) is the table whose elements (rows) consist of exactly those rows *r* of *x* for which *p*(*r*) = true.

Note. This would be a defined-type, but the type-declaration syntax does not permit the parameter-list to be a variable-length list of field-specifiers.

Issue: 3

Subject: Reword multiple repertoire explanation

Type: E

Clauses: 8.1.4

Xref:

Rationale: poor wording

Proposed Change:

Change “All repertoire identifiers ... character-set.” to “All repertoire identifiers in a given



repertoire list shall designate subsets of the same reference character-set.”

Change “only the distinct members” to “the (set) union of the character-sets (without duplication)”.

Issue: 4

Subject: Make Ordinal a defined-type

Type: minor

Clauses: 8.1.5

Xref: X3T2/93-006

Rationale: Ordinal can be completely and correctly described in a type-declaration based on integer and such a declaration eliminates an existing ambiguity in the value syntax. (Is “1” an ordinal value or an integer value?)

Proposed Change: Move the definition of the ordinal type from clause 8.1 to clause 10.1, replacing the “syntax” paragraph with:

Declaration: type ordinal = new integer: range(1..\*); .

Issue: 5

Subject: Make Bit a defined-type

Type: minor

Clauses: 8.1.7

Xref: X3T2/93-006

Rationale: Datatype Bit is precisely Modulo(2) and should be so declared.

Proposed Change: Move the definition of the bit datatype from clause 8.1 to clause 10.1, replacing the “syntax” paragraph with:

Declaration: type bit = modulo(2); .

Issue: 6

Subject: Inappropriate characterizing operations for choice

Type: minor

Clauses: 8.3.1

Xref: WG11-2.9

Rationale: The choice-type, as currently described, is one of two useful models of the choice-type and is that which maps most readily to programming languages and to existing RPC implementations. However, the value model (tag-value, alternative-value) is not supported correctly by the characterizing operations.

Proposed Change:

a. Replace the IsType operation with:

Discriminant(x: choice (tag-type) of (alternative-list)): tag-type is the tag-value of the value x;

b. Rephrase the definition of Cast as:

If the tag-value of x selects an alternative whose alternative-type is type, then that value of type which is the alternative-value of x, else undefined;

c. Rephrase the definition of Equal as:

If Discriminant(x) and Discriminant(y) select the same alternative, then type.Equal(Cast.-type(x), Cast.type(y)), where type is the alternative-type of the selected alternative and type.Equal is the Equal operation on datatype type, else false.

d. Add

Note: the operation Discriminant is a conceptual operation which reflects the ability to determine which alternative of a choice-type is selected in a given value. Any equivalent mecha-

nism is satisfactory. When a choice-value is moved between two contexts, as between a program and a data repository, representation of the chosen alternative is required, and most implementations explicitly incorporate the tag-value.

Note. The other useful model is choice(field-list) where exactly one field is present in any given value and the means of discrimination is not specified. In this model,

IsField.field(x: choice(...)): boolean = true if the designated field is present in the value x, replaces Discriminant, with corresponding changes to the other characterizing operations. It is recognized that this model is mathematically more elegant (the Or-graph to match the And-graph of the fields in Record), but in practice, either IsField is not provided (which makes all operations user-defined) or IsField is implemented by tag-value (which makes IsField equivalent to Discriminant).

#### Issue: 7

Subject: Relationship of pointers to boxes

Type: E

Clauses: 8.3.2

Xref: X3T2/92-112

Rationale: Many comments suggest that the relationship between the LID pointer datatype and the language notion "variable" and the implementation notion "memory cell" are confused.

Proposed Change: Add the following Note:

Note – The hardware implementation of the "means of reference to" a value of the element-type is usually a memory cell or cells which contain a value of the element-type. The memory cell has an "address" which is the "value of the unspecified state datatype". The memory cell physically maintains the association between the "address" (pointer-value) and the element-value which is stored in the cell. The Dereference operation is conceptually applied to the "address" (pointer-value) but is implemented by a "fetch" from the memory cell. Thus in the computational model used here, the "address" and the "memory cell" are not distinguished: a pointer-value is both the cell and its address, because the cell can only be manipulated through its address. The cell, which is the pointer-value, IS distinguished from its contents, which is the element-value. The notion "variable of datatype T" appears in programming languages and is usually implemented as a cell which contains a value of type T. Language standards often distinguish between the "address of the variable" and the "value of the variable" and the "name of the variable", and one might conclude that the "variable" is the cell itself. That conclusion is not useful. In the computational model used here, the "address of the variable" is a value of LI datatype "pointer to (T)", while the "value of the variable", meaning the contents of the cell, is a value of LI datatype T, and all operations on the "variable" must operate on one of those two values. This notion is further elaborated in DIS nnnn: LIPC, which relates pointer-values to the "boxes" (or "cells"), which are elements of the state of a running program.

#### Issue: 8

Subject: Remove Switch

Type: minor

Clauses: 10.1.1

Xref: X3T2/93-006

Rationale: The syntax for type-declarations no longer supports Switch, and it seems to be nothing more than a renaming of boolean and its values.

Proposed Change: Delete clause 10.1.1.

## Issue: 9

Subject: Rename "Cardinal" "NaturalNumber".

Type: minor

Clauses: 10.1.2

Xref: X3T2/93-006

Rationale: The term natural number appears several times in CD 11404, while the term "cardinal" never does. While "cardinal" is one proper mathematical term, "natural number" is equally acceptable in mathematics and much more common in computer science, information analysis, etc.

Proposed Change: Change all occurrences of "Cardinal" to "NaturalNumber".

## Issue: 10

Subject: Provide guidance on "identifier" datatypes and "syntactic strings"

Type: minor

Clauses: 10.1.4

Xref: X3T2/93-006

Rationale: Languages like LISP and Prolog which have dynamic evaluation of certain kinds of "syntactic strings" think of these as having a datatype. Many special-purpose languages have "variables" or "values" which are represented as character-strings, but have important semantic properties. It appears that such objects must be character-strings when manipulated by other languages.

Proposed Change: Add a Note to clause 10.1.4 (Character-string) showing an example of declaring a "character-string" datatype which has important semantic properties, as was done for Prolog (in Tampere).

## Issue: 11

Subject: Remove "Currency"

Type: minor

Clauses: 10.1.6, 8.1.10

Xref: X3T2/93-006

Rationale: This type definition is only appropriate to some major international currencies. It is inappropriate for Japan, Italy, Spain, Mexico, to name but a few. An example for some appropriate currency should appear under Scaled.

Proposed Change:

a. Remove clause 10.1.6.

b. Add Example: type dollars = scaled(10,2); with appropriate explanation in 8.1.10.

## Issue: 12

Subject: Rename "Interval"

Type: minor

Clauses: 10.1.7

Xref: X3T2/93-006

Rationale: "Interval" is a valid datatype in Ada, Modula and many real-time applications regardless of language. But the meaning is "time-interval" not "distance".

Proposed Change: Replace all occurrences of "Interval" with "TimeInterval".

**Issue: 13**

**Subject:** Remove “distinguished name”

**Type:** minor

**Clauses:** 10.1.11

**Xref:** X3T2/93-006

**Rationale:** RPC asked for this datatype but it does not appear in the RPC standard. While it is apparently defined by a recently adopted ISO standard (Directory Services), it is only one of many ISO standard naming syntaxes. It’s inclusion in LID does not seem to meet any immediate need.

**Proposed Change:** Remove clause 10.1.11.

**Issue: 14**

**Subject:** Remove cyclic-enumerated

**Type:** minor

**Clauses:** 10.2.3

**Xref:** X3T2/93-006

**Rationale:** This datatype is not common in the sense that it appears in many (any?) programming languages or many interfaces. Moreover, it serves only to modify a characterizing operation of enumerated-types and that very slightly.

**Proposed Change:** Delete clause 10.2.3.



Second

VOTE ON COMMITTEE DRAFT <del>11404</del> 11404	
Date of circulation 1993-01-09	Reference number
Closing date for voting 1993-04-16	ISO/TC1 /SC22 N/305

ISO/TC 1 /SC 22
Title Programming Languages, their environments and system software interfaces
Secretariat Canada, SCC

Circulated to P-members of the committee for voting on registration of the draft as a DIS, in accordance with 2.4.3 of part 1 of the IEC/ISO Directives

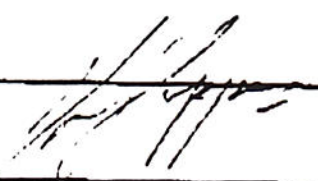
Please send this form, duly completed, to the secretariat indicated above.

CD 11404
Title Information Technology - Programming Languages, their environments and system software interfaces - Language-Independent Datatypes

- We agree to the circulation of the draft as a DIS in accordance with 2.5.1 of part 1 of the IEC/ISO Directives
- We do not agree to the circulation of the draft as a DIS  
The reasons for our disagreement are the following (use a separate page as annex, if necessary)

The absence of the value **UNDEFINED** in the domain of all types specified in this draft is regretted. It is considered that while few current languages have such a requirement, such a requirement will become much more common in the future as greater use is made of formal methods in all aspects of Information Technology.

While the need for a concrete syntactic form for the draft is obvious, it is surprising that an abstract syntax has not been defined so that proof of equivalence in the mapping from language definitions/bindings can be made more readily.

P-member voting	<b>NEW ZEALAND</b>	
Date	16 April 1993	Signature

