ISO/IEC JTC1/SC22
Languages
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC22
N 906

JANUARY 1991

TITLE: Summary of Voting on: Proposal to register N842 - Programming Languages Common Language-independent Datatypes as a Committee Draft (CD)

SOURCE: Secretariat ISO/IEC JTC1/SC22

WORK ITEM : JTC1.22.17

STATUS : New

CROSS REFERENCE : N842

DOCUMENT TYPE : Summary of Voting

ACTION : For information to SC22 Member Bodies.
See Attached.

Address reply to: ISO/IEC JTC1/SC22 Secretariat
J.L. Côté, 140 O'Connor St., 10th Floor
Ottawa, Ont., Canada K1A 0R5
Telephone: (613)957-2496 Telex: 053-3336
Fax: (613) 996-2690

SUMMARY OF VOTING ON:

Letter Ballot Reference No: SC22 N842
Circulated by :JTC1/SC22
Circulation Date :1990-09-28
Closing Date :1990-01-08

SUBJECT: Proposal to register N842 as a Committee Draft (CD)
Common Language-Independent Datatypes

The following responses have been received:

'P' Members supporting the proposal
without comments : 08 See Attached List

'P' Members supporting the proposal,
with comments : 01 See Attached List

'P' Members not supporting the proposal: 01 See Attached List

'P' Members abstaining :00

'P' Members not voting: 09 (see list)

Attachment 1 - France's Comments
Attachment 2 - USA's Comments

Secretariat Action:

The comments received have been submitted to WG11 for consideration. Based on WG11's recommendation, document N842 or a revised version will be registered as a Committee Draft.

ISO/IEC JTC1/SC22 LETTER BALLOT SUMMARY

PROJECT NO: JTC1.22.17

SUBJECT: Proposal to register N842 as a Committee Draft(CD) Common Language-Independent Datatypes

Reference Document No: N842
Circulation Date: 1990-09-28

Ballot Document No: N842
Closing Date: 1991-01-08

Circulated To: SC22 P,O,L

Circulated By: Secretariat

SUMMARY OF VOTING AND COMMENTS RECEIVED

	Approve	Disapprove	Abstain	Comments	Not Voting
'P' Members					
Austria	( )	( )	( )	( )	( )
Belgium	(x)	( )	( )	( )	( )
Canada	(x)	( )	( )	( )	( )
China	( )	( )	( )	( )	( )
Czechoslovakia	( )	( )	( )	( )	( )
Denmark	(x)	( )	( )	( )	( )
Finland	( )	( )	( )	( )	( )
France	(x)	( )	( )	(x)	( )
Germany	( )	( )	( )	( )	( )
Iran	( )	( )	( )	( )	( )
Italy	(x)	( )	( )	( )	( )
Japan	(x)	( )	( )	( )	( )
Netherlands	(x)	( )	( )	( )	( )
New Zealand	( )	( )	( )	( )	( )
Sweden	( )	( )	( )	( )	( )
Switzerland	( )	( )	( )	( )	( )
UK	(x)	( )	( )	( )	( )
USA	( )	(x)	( )	(x)	( )
USSR	(x)	( )	( )	( )	( )
'O' Members					
Australia	( )	( )	( )	( )	( )
Brazil	(x)	( )	( )	( )	( )
Hungary	( )	( )	( )	( )	( )
Icelan	( )	( )	( )	( )	( )
India	( )	( )	( )	( )	( )
Korea	( )	( )	( )	( )	( )
Norway	( )	( )	( )	( )	( )
Poland	( )	( )	( )	( )	( )
Portugal	( )	( )	( )	( )	( )
Singapore	( )	( )	( )	( )	( )
Turkey	( )	( )	( )	( )	( )
Thailand	( )	( )	( )	( )	( )
Yugoslavia	( )	( )	( )	( )	( )

Attachment 1 to N806

AFNOR COMMENTS ON ISO/IEC JTC1/SC22/N842

COMMON LANGUAGE INDEPENDENT DATATYPES,

AND ISO/IEC JTC1/SC22/WG11/N205

General

1 The document attempts to fit to a strictly mathematical model. However, some of the notions used, such as reference, instance, undefined, null, pointer, bag, value, ... are data processing notions for which the mathematical model is not sufficient. The model used should be presented.

Page i, Issue 3

2 Pragmata should remain in the document. They are a "standardized" way to express mappings, e.g. those required from CLID to/from language datatypes.

Page i, Issue 4

3 Mappings should remain in the document as a normative part.

Page iii, Foreword

4 The first paragraph refers to PHIGS. A footnote or a reference should tell what it is.

5 In the fourth line of the fifth paragraph, the datatypes should read some of the datatypes.

Clarification of what is part of the Standard

6 NOTEs do not seem to be part of the Standard. This should explicitly be stated at the beginning of the document.

7 Annexes (appendices) are usually not part of the Standard. There is no reason for Annexes A, B, and C to be annexes since they are declared as normative; the references to these annexes in Section 5, Compliance, could as well be done to plain Sections, rather than to Annexes.

Page 1, 1. Scope

8 In the first paragraph, last line, a space is missing between *datatype* and *are*.

9 The last sentence in the fourth paragraph needs to be more clear or explicit to be understood.

Pages 2 to 4, 3. Definitions

10 Definitions are needed for *mapping*, *generic mapping*, *generic datatype*, *primitive internal datatype*, and *generated internal datatype*.

Page 4, 4.1. Formal syntax

11 It should be explained that the use of the BNF is not only applicable to datatype identification, but also to datatype values. It should also be explained to which extent it can be used for datatype values (primitive datatype values?, more?) and why it is not used beyond, or why it does not need to be used beyond.

Page 5, 4.1. Formal syntax

12 On the fifth line of the first paragraph of the page, construction needs to be explained.

Page 5, 4.2. Lexical objects

13 **Any-character** is not defined (last rule).

Page 5, 5. Compliance

14 Examples/illustrations of what is meant by the various sorts of compliance would help understanding.

Page 6, 5.1. Direct compliance

15 **The usefulness of the concept of direct compliance is questionable. What language, service, ... can comply directly to the required set of datatypes, specially since some of those required datatypes are defined with infinite value spaces.**

Pages 6 and 7, 5. Compliance

16 **The user (language, service, ...) can define new (extra) datatypes. However, when two users have defined their own set of extra datatypes, the communication between them may be difficult if they use these extra datatypes, though maybe both users totally comply to the CLID standard. In other words, there is a difference between those who provide an extra datatype and those who use an extra datatype. Should not this problem be addressed in the compliance section?**

17 **Establishing, as in annex A, the list of required datatypes for compliance excludes some well known languages (COBOL does not support booleans so far). The rule should not require from languages, for them to be compliant, to support datatypes that may be outside their scope. However, the rule might require from languages to document what they support and what they do not support.**

Page 7, 6.1. Datatype

18 **The use of the word "value" is unusual; it seems that the right word should be element (or maybe object). (See comment 1)**

19 **The rationale for characterizing operations in the definition of a datatype needs to be given. In fact, it seems that there is no need for characterizing operations, and that they should be withdrawn. The only important thing is that datatypes have an unambiguous definition. For example, the datatype *real* is the set of real numbers in the mathematical sense; the datatype *list of real* is the set of sequences of real numbers.**

If the definition of CLI datatypes is unambiguous, then it is up to the language designers to define the mappings between CLI datatypes and their internal (language) datatypes. Going on with the above example, the operations add or multiply are not used at all in the definition of the mapping, and the mapping could be defined even if the language did not support the operations. In addition, not having characterizing operations would greatly simplify the document, and make it easier to understand.

Page 8, 6.3.1. Equivalence

20 Is the equivalence notion defined in every value space the same as the notion of identity between two objects. If so, it should be stated. The one exception that can be seen is that of rational; if the value space is the set of couples of numbers, then 2/3 is not identical to 4/6, but the two rational numbers that they represent are identical.

Note that reading the NOTE in Page 9, 6.3.3. Bound, it seems that equality implies unicity (U1 = U2 implies that the upper bound is unique).

Page 10, 6.4.2. Dense and discrete, NOTE 1

21 There is an infinite number of rationals between two rationals; this does not prevent any rational from having a computer representation (problem of size).

Page 12, 7.1. Primitive datatypes

22 The *file* datatype does not exist, but should. It is not only a collection of records. It has some attributes related to the container, as for example: the organization, the record keys, the record type, the padding characters, the record delimiters, the position indicator, ... A *file* can also be pointed to, assigned to, passed as a parameter, as a *procedure* can.

23 The purpose of the syntax paragraph (the syntax productions for the datatype) should be clarified, i.e. it does not define the set of values, but the "designator" of the datatype.

24 **The notation used in the operations paragraph should be explained; for example, looking below at Boolean Operations, Or(x,y:boolean):boolean without explanation of the notation requires some time to understand.**

Page 14, 7.1.2. State

25 **Under Syntax, add the rule value-identifier = identifier.**

26 **The metavariable parameter-list is only defined in Annex F (which is informative!). This comment is applicable to many other datatypes.**

27 **The text under Operations is only equivalence. First, equivalence is not an operation, but a relation; Equal is the operation. In addition, since the paragraph 6.3.1. specifies that this operation is defined for every datatype, there is no more need to repeat it here than for the other datatypes.**

Page 14, 7.1.4. Character

28 **An example is needed to illustrate the relation between a character and the *alphabet-identifier*, how a character-value can be shown by its *positional-value*, how a user-defined alphabet can be accommodated.**

Pgae 15, 7.1.5. Ordinal

29 **Is it really necessary to distinguish between Ordinal and Integer? This comment is also related to the comment relative to characterizing operations.**

Page 16, 7.1.6. Date-and-Time

30 **Another example of the distinction between denotation (date-and-time-value) and the denoted object (a point in time).**

31 **The notation Extend.*resltores2* ... needs to be explained.**

Page 17, 7.1.8. Integer
Page 18, 7.1.10. Scaled

32 Can a FORTRAN integer be mapped on a *scaled(2,0)* CLI datatype?

33 Can a COBOL PIC 9(n) be mapped on an *integer* CLI datatype?

Page 18, 7.1.10. Scaled

34 If for example the radix is 16, is the definition of **scaled-value** proper since it is given in terms of numbers, i.e. of decimal digits?

Page 19, 7.1.10. Scaled

35 It seems that NOTE 3 contradicts the last paragraph of 7.2. Subtypes, on page 25.

Page 19, 7.1.11. Real

36 There seems to be a contradiction between the informal description of Real (the set of real numbers in the mathematical sense), and the syntactical definition of real-value (which is given in terms of decimal digits, thus not allowing all radixes). In addition, number permits any number of digits, but not an infinite number of digits. One can however wonder whether it is desirable to define datatypes that imply an infinite number of digits, and therefore that are not capable of being represented in a computer.

Page 21, 7.1.12. Complex

37 Measure(x,y:complex):real is the distance of the points x and y; what is meant by ??? ?

Page 21, 7.1.13. Null

38 The Description states that **Null** represents an object; a datatype does not represent an object; it is a collection of objects.

39 *Null* should be a value of the value space of each datatype. It would have the effect that an operation that is valid on any other value of the datatype fails with this special *null* value. Only equivalence works, inside a given datatype.

40 The question on whether this *null* value only applies to primitive datatypes, or applies to any datatype, is to be decided. For example: what is a *null list*? can a non *null* list contain a *null* element?

41 Obviously a *pointer*, which is not a primitive datatype, can be *null*. But is a *pointer* a non primitive datatype?

Page 21, 7.1.14. Undefined

42 The Description states that Undefined represents an object; a datatype does not represent an object; it is a collection of objects.

43 *Undefined* should be a special value of the value space (including *null*) of each datatype, or a status of the association of a datatype with its representation. It would have the effect that the result of an operation that is normally valid on the datatype, becomes unpredictable (including the fact that the operation may fail if the actual value happen to be *null*).

44 A non primitive datatype may be *undefined*, with the effect that each of its related element is *undefined*. Note however that for a non primitive datatype to be *undefined*, it is sufficient that one of its base elements is *undefined*, which seems to suggest two sorts of *undefinitions* for non primitive datatype.

45 What is the meaning for the related element of an non-*null undefined pointer*?

Page 22, 7.1.16. Procedure

46 The term terminating functions used in the first line of the Values paragraph does probably mean terminating algorithms. Maybe the definition should be reworked.

Page 29, 7.3.1. Choice

47 In the **Syntax** paragraph, are the datatypes of the **datatype-list** within the **alternative-list** disjoint?

48 In the **Operations**, **Cast.*alternative*** needs clarification.

Page 30, 7.3.2. Record

49 In the **Description** paragraph, change **aggregations of values** to **aggregations of named values**.

Page 31, 7.3.3. Pointer

50 In some languages, as for example PL/I, a pointer is not related to a specific datatype. Is the mapping of such pointers to CLID of the form *pointer to choice of any*?

51 Is not the nature of pointers *primitive*?

Page 33, 7.3.4. Set
Page 34, 7.3.6. Bag

52 **Select** is not an operation in the usual sense. Should not it be deleted?

Page 41, 8.3. Value Declarations

53 There are not value-notations for all values; for example, there is no list-value.

Page 49, B.3. Bit string
Page 49, B.4. Character string

54 These datatypes are defined in terms of *list*, but the operations are *First*, *Rest*, ... instead of *Head*, *Tail*, ...

55 Why is the concept of *limits* as shown in *list* not kept in *bit string* and *character string*. Should not they be defined as *new list [(limits)] of bit/character*?

Page 59, D.8. Alignment

56 Some languages specify that a representation is **both synchronized left and right. It is suggested that the definition of *sync-point* be changed to:**

sync-point = "left" | "right" | "both"

with the appropriate explanations.

Page 62 to 65, Annex F, Collected Syntax

57 Some production rules are missing, e.g.: **letter, digit, hyphen, special, space, any-character.** In the rule for **value-notation, time-value** should read **date-and-time-value.**

Improvement in the readibility of the document

58 There is a need for an unformal presentation of **the technical material, a kind of primer** or rationale, that could be shown as an appendix.

59 **Examples are necessary; more particularly** some examples of mappings CLID/Languages would help those language committees who will have to attempt the exercise of writing such mappings.

Miscellaneous

60 Before it proceeds to the next stage in the ISO approval cycle, it is needed that the document be used to write a complete mapping of a standardized language/interface from/to CLID. This would allow to assert its applicability in the real world.

attachment 2
to N906

U.S. Position on ISO/IEC JTC1/SC22/N842
CLIDT Committee Draft Registration

The U.S. votes NO on the SC22 letter ballot to register SC22/WG11's Working Draft #4 on Programming Languages Common Language-Independent Datatypes as a Committee Draft for the following reason.

- In the procedures for the technical work of ISO/IEC Joint Technical Committee 1 (JTC 1) on Information Technology, section 6.5.1.1 states:

> "The WD can be considered as having reached the stage of committee draft when:
>
> the main elements have been included in the document;
>
> it is presented in a form which is essentially that envisaged for the future International Standard;
>
> it has been dealt with at least once by JTC 1 or by a working body of JTC 1;
>
> JTC 1 or one of its SCs has decided in a resolution during a meeting or by letter ballot that the WD be forwarded to the ITTF for registration as a CD

It is the position of the U.S. that in the case of the CLIDT, it does not meet all of the requirements as dictated by the JTC 1 procedures. The CLIDT contains a number of significant outstanding issues which when resolved may alter the contents of the CLIDT dramatically. Therefore, the U.S. requests that the Outstanding Issues section be removed from the document and that a majority of the substantive issues be resolved prior to the CLIDT being registered as a Committee Draft.

Proposed Changes

- Major Technical: Array should be added as a datatype generator in Annex A.
- Major Technical: The Null datatype should be removed from the CLIDT draft. The concept of Null can be replaced with a more general concept called "Status". Each datatype would then consist of the union of well defined distinct values and the collection of distinct statuses. Each status would then represent an exceptional condition associated with that datatype. Some examples of types of statuses would be Null, Undefined, and Out of Range. The addition of status will also allow the CLIDT to close all undefined operations by utilizing this concept. The subtyping mechanism needs to be extended to deal with status in addition to values.
- Major Technical: Null should be removed from Annex A as a required primitive datatype.
- Major Technical: The mappings clause is appropriate for the CLIDT standard. This section should define a mapping model that contains the requirements and guidelines for mappings.
- Major Technical: The CLIDT syntax should provide hooks in the CLIDT syntax in order to provide additional annotations. The CLIDT should define an annotation mechanism that can be used by applications such as Remote Procedure Call and the Common Language-Independent Procedure Calling Mechanism. A new issue in this area is whether or not the CLIDT should define a module concept.

The framework for modules would include issues such as scope and usage. Annotations at the module level should also be considered.

- Major Technical: Pointer datatypes should support a Null status.
- Major Technical: An additional subtype generator should be defined to allow the creation of new datatypes by the explicit exclusion of values or statuses either by enumeration or as ranges. For example, you can create a subtype from Choice for which Null is not a status.
- Editorial: Change the definition of "primitive datatype" in section 3.32 to the following:

> 3.32 primitive datatype: an identifiable datatype that cannot be decomposed into other identifiable datatypes without loss of all semantics associated with the datatype.

- Editorial: The following definitions should be added to the CLIDT:

> data interchange format: a standard form of representing data of various datatypes for the purpose of transferring data between applications, subprograms, or machines.
>
> datatype identifier: a standard name or other syntactic construction which uniquely identifies a particular primitive datatype or non-primitive datatype.