INTERNATIONAL STANDARD

ISO/IEC 8824-3

Third edition
2002-12-15

Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification

Technologies de l'information — Notation de syntaxe abstraite numéro un (ASN.1): Spécification des contraintes

Reference number
ISO/IEC 8824-3:2002(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published by ISO in 2003

Published in Switzerland

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 8824-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.682.

This third edition cancels and replaces the second edition (ISO/IEC 8824-3:1998), which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 8824-3:1998/Cor.1:2001, ISO/IEC 8824-3:1998/Cor.2:2002 and ISO/IEC 8824-3:1998/Cor.3:2002.

ISO/IEC 8824 consists of the following parts, under the general title *Information technology — Abstract Syntax Notation One (ASN.1)*:

— *Part 1: Specification of basic notation*

— *Part 2: Information object specification*

— *Part 3: Constraint specification*

— *Part 4: Parameterization of ASN.1 specifications*

Introduction

Application designers require a notation to define a structured data type to convey their semantics. This is provided in ITU-T Rec. X.680 | ISO/IEC 8824-1 and ITU-T Rec. X.681 | ISO/IEC 8824-2. A notation is also required to further constrain the values that can appear. Examples of such constraints are restricting the range of some component(s), or using a specified information object set to constrain an "ObjectClassFieldType" component, or using the "AtNotation" to specify a relation between components.

This Recommendation | International Standard provides the notation for the general case of constraint specification.

NOTE 1 – For historical reasons the special case of a "subtype constraint" is specified in ITU-T Rec. X.680 | ISO/IEC 8824-1.

Constraint notation can appear (in round brackets) after any use of the syntactic construct "Type", and the purpose of this Recommendation | International Standard is to specify the general case of what goes in the round brackets.

NOTE 2 – Multiple constraints (each inside its own round brackets) can be applied to the same "Type", as the result of constraining a "Type" is itself formally a "Type" construct.

When a constraint is applied to the textually outermost use of a "Type" construct, it results in the creation of a new type which is a subtype of the original (parent) type.

A subtype of a parent type can itself be used in defining other subtypes of the same parent type in other uses of the constraint notation. Thus the subset of values constituting a subtype can be defined either by limiting the range of the parent type, or by specifying the subtype as a union of sets of values.

NOTE 3 – The "ValueSet" notation specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 15.7, provides a further means of specifying a subtype.

Constraints may also be used to produce a subtype of a parent type (as described above) when the notation is embedded within another type. However, some "component relation" constraints are textually included following a "Type" (within a set or sequence type definition), but are not used to restrict the set of possible values of the "Type" which they follow (the referencing component). Rather, they specify a relation between the value of the referencing component and the value of one or more other "Type"s in the same set or sequence type (called the referenced components).

Component relation constraints can be seen as subtyping the sequence type within which they are embedded, but not necessarily the referencing type.

A constraint on an "ObjectClassFieldType" component can be applied by restricting the type or values in the component by using an information object set. Such constraints are called table constraints, since they are specified in terms of the "associated table" of the object set. The component relation constraints defined in this Recommendation | International Standard are a special case of table constraints.

Finally, a "Type" may be subtyped by specifying the set of values in the subtype by human-readable text. Such a constraint is called a user-defined constraint. For example, a user-defined constraint can be specified to constrain a **`BIT STRING`** to the set of values produced by the encryption of a value of a specified ASN.1 type.

It is the purpose of this Recommendation | International Standard to provide the notation to be used for specifying table constraints (including component relation constraints), and user-defined constraints.

NOTE 4 – In general, full support for the specification of constraints in a flexible way (particularly component relation constraints, subtyping constraints, and user-defined constraints with a formally defined body) would require notation with a power comparable to that of programming languages. Such power can only be sensibly provided by the establishment of links from the ASN.1 notation into some other defined computer language. This version of this Recommendation | International Standard does not provide such links, and hence supports only a small number of constraining mechanisms.

While the embedding of notation defining constraints (subtypes and relationships) will frequently be the most convenient form of specification (particularly for the simple subtyping of primitive components of structures), separate (external) specification will sometimes be preferred, particularly where the constraints are being imposed by a separate group from that which defined the basic protocol.

NOTE 5 – The parameterization defined in ITU-T Rec. X.683 | ISO/IEC 8824-4 is specifically designed to enable a piece of ASN.1 specification (and in particular, a constraint) to be parameterized, allowing the actual constraint to be imposed by some other group that provides actual parameters for the parameterized construct.

The notations for constraint specification supported here are:

- user-defined constraints (see clause 9);
- table constraints, including component relation constraints between two components which are carrying values related to an information object, defined using the notation of ITU-T Rec. X.681 | ISO/IEC 8824-2 (see clause 10);
- contents constraints (see clause 11).

The application of table constraints to the "InstanceOfType" construct of ITU-T Rec. X.681 | ISO/IEC 8824-2, Annex C, is specified in Annex A.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification

1 Scope

This Recommendation | International Standard is part of Abstract Syntax Notation One (ASN.1) and provides notation for specifying user-defined constraints, table constraints, and contents constraints.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 Specification of basic notation

This Recommendation | International Standard uses the terms defined in ITU-T Rec. X.680 | ISO/IEC 8824-1.

3.2 Information object specification

This Recommendation | International Standard uses the terms defined in ITU-T Rec. X.681 | ISO/IEC 8824-2.

3.3 Parameterization of ASN.1 specification

This Recommendation | International Standard uses the following term defined in ITU-T Rec. X.683 | ISO/IEC 8824-4:

- parameterized type.

3.4 Additional definitions

3.4.1 component relation constraint: A constraint on the values of a set type or sequence type which is textually associated with one of the component types (the referencing component) of the set type or sequence type, and which

specifies the relationship between the value of that component and the values of one or more other components (the referenced components).

3.4.2 constrained type: The innermost "Type" which contains the referencing component and all of the referenced components of some component relation constraint.

3.4.3 constraining set: The information object set referenced in some component relation constraint.

3.4.4 constraining table: The associated table (see ITU-T Rec. X.681 | ISO/IEC 8824-2, clause 13) corresponding to a constraining set.

3.4.5 referenced component: A component of a set type or sequence type identified in a component relation constraint.

3.4.6 referencing component: A component of a set type or sequence type which has an associated component relation constraint.

3.4.7 selected rows: Those rows of a constraining table which contain, in the appropriate columns, the values of all of the referenced components.

3.4.8 table constraint: A constraint applied to an object class field type which demands that its values conform to the contents of the appropriate column of some table.

3.4.9 user-defined constraint: A constraint which requires a more complicated statement than can be accommodated by the other forms of constraint, and which must therefore involve specification by some means outside of ASN.1.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviation applies:

ASN.1 Abstract Syntax Notation One

5 Convention

This Recommendation | International Standard employs the notational convention defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 5.

6 Notation

This clause summarizes the notation defined in this Recommendation | International Standard.

6.1 Constraint

The following notation which can be used as an alternative for "ConstraintSpec" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 45.6) is defined in this Recommendation | International Standard:

– GeneralConstraint (see 8.1).

7 ASN.1 lexical items

In addition to the lexical items specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 11, this Recommendation | International Standard makes use of the lexical items specified in the following subclauses. The general rules applicable to these lexical items are as defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.1. These new lexical items make use of the ASN.1 character set, as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 10.

7.1 Additional keywords

The names **CONSTRAINED**, **CONTAINING**, **ENCODED** and **BY** are listed in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.27, as reserved words.

8 General constraint specification

8.1 The notation for a "GeneralConstraint" is as follows:

```
GeneralConstraint ::=
        UserDefinedConstraint
    |   TableConstraint
    |   ContentsConstraint
```

8.2 The various possibilities for specification of the constraint are defined as follows:

a) "UserDefinedConstraint", in clause 9;

b) "TableConstraint", in clause 10;

c) "ContentsConstraint", in clause 11.

9 User-defined constraints

NOTE 1 – This form of constraint specification can be regarded as a special form of ASN.1 comment, since it is not fully machine-processable. However, it would be possible for an automatic tool to use the presence of a particular user-defined constraint to invoke user-supplied constraint checking.

NOTE 2 – Protocol designers should be aware that since the definition of a constraint in this way is not fully machine-processable, a specification which employs this capability may be less easy to handle with automatic tools.

9.1 A user-defined constraint is specified by the syntax:

```
UserDefinedConstraint ::=
    CONSTRAINED BY "{" UserDefinedConstraintParameter "," * "}"
```

9.2 It is recommended that the actual constraint be referenced by a comment anywhere inside the braces ("**{**" and "**}**"). This comment should clearly state what constraint is imposed by the "UserDefinedConstraint".

NOTE – If there are any "UserDefinedConstraintParameter"s within the braces (see 9.3), the comments may precede, follow, or be interspersed among them, at the definer's convenience.

9.3 The actual constraint to be applied may depend on some parameters. For each such parameter, a "UserDefinedConstraintParameter" shall be included in the "UserDefinedConstraint". Each "UserDefinedConstraintParameter" shall be any "Value", "ValueSet", "Object", "ObjectSet", "Type" or "DefinedObjectClass" which is defined inline or is a reference name.

NOTE – The reference name may be a dummy parameter if the "UserDefinedConstraint" is used within a "ParameterizedAssignment".

```
UserDefinedConstraintParameter ::=
        Governor ":" Value
    |   Governor ":" ValueSet
    |   Governor ":" Object
    |   Governor ":" ObjectSet
    |   Type
    |   DefinedObjectClass
```

The notation "Governor" is defined in ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.3. When the first or second alternatives are used, the "Governor" shall be a "Type". When the third or fourth alternatives are used, the "Governor" shall be a "DefinedObjectClass".

9.4 Example

If an application designer wishes to specify that certain components are to be bit strings carrying an encryption of the value of some ASN.1 type (different for each component), then (using the parameterization of ITU-T Rec. X.683 | ISO/IEC 8824-4) the parameterized **ENCRYPTED** type can be defined as follows:

```
ENCRYPTED {ToBeEnciphered} ::= BIT STRING
      (CONSTRAINED BY
          {-- must be the result of the encipherment of some BER-encoded
           -- value of -- ToBeEnciphered}
      ! Error : securityViolation)
```

```
Error ::= ENUMERATED {securityViolation}
```

and a use of the **ENCRYPTED** parameterized subtype of **BIT STRING** (which is what the **ENCRYPTED** type is) becomes simply:

```
ENCRYPTED{SecurityParameters}
```

or, equivalently, at the whim of the designer:

```
BIT STRING (ENCRYPTED{SecurityParameters})
```

The occurrence of a **securityViolation** is handled according to local security policy.

10 Table constraints, including component relation constraints

NOTE 1 – Information object classes, information objects, information object sets, and the object class field type are defined in ITU-T Rec. X.681 | ISO/IEC 8824-2. An understanding of these concepts is assumed in this clause.

NOTE 2 – This clause describes the application of the table constraint using an information object set that is identified within the main notation defining the parent type, in other words, defined and identified by the protocol designer. This does not satisfy the requirement for the actual information object set which is to be used as the constraint in particular abstract syntaxes to vary from syntax to syntax. ITU-T Rec. X.683 | ISO/IEC 8824-4 provides notation which, among other things, enables the information object set used in this constraint to be a parameter whose value is supplied at a later date by varying groups.

Example

For the purpose of illustrating the text of this clause, the following example will be used. An **ErrorReturn** type carries an **errorCategory** and one or more **errorCodes** with corresponding **errorInfo** from that category. This is supported by an **ERROR-CLASS** information object class with a specific set of objects defined in the information object set **ErrorSet** that is used to constrain the fields of **ErrorReturn**.

We have:

```
ERROR-CLASS ::= CLASS
      {
      &category  PrintableString (SIZE(1)),
      &code INTEGER,
      &Type
      }
      WITH SYNTAX {&category &code &Type}

ErrorSet ERROR-CLASS ::=
      {
      {"A" 1 INTEGER} |
      {"A" 2 REAL} |
      {"B" 1 CHARACTER STRING} |
      {"B" 2 GeneralString}
      }

ErrorReturn ::= SEQUENCE
      {
        errorCategory        ERROR-CLASS.&category ({ErrorSet}) OPTIONAL,
        errors               SEQUENCE OF SEQUENCE
        {
          errorCode  ERROR-CLASS.&code ({ErrorSet}{@errorCategory}),
          errorInfo  ERROR-CLASS.&Type ({ErrorSet}{@errorCategory,@.errorCode})
        } OPTIONAL
      }
```

The associated table of **ErrorSet** can be depicted as follows:

&category	&code	&Type
"A"	1	INTEGER
"A"	2	REAL
"B"	1	CHARACTER STRING
"B"	2	GeneralString

10.1 A table constraint can only be applied to types "ObjectClassFieldType" or an "InstanceOfType". The former case is defined in the remainder of this clause, the latter in Annex A.

10.2 An "ObjectClassFieldType" identifies an information object class, and one of the permissible "FieldName"s of that class. The table constraint identifies the set of information objects whose associated table (as defined in ITU-T Rec. X.681 | ISO/IEC 8824-2, clause 13) determines the set of constrained values.

10.3 The "TableConstraint" notation is:

TableConstraint ::=
SimpleTableConstraint |
ComponentRelationConstraint

SimpleTableConstraint ::= ObjectSet

10.4 The "ObjectSet" in the "SimpleTableConstraint" is governed by the class which appears in the "ObjectClassFieldType" being constrained.

10.5 The semantics of the "SimpleTableConstraint" are specified using the associated table of the constraining information object set.

10.6 The "FieldName" of the type being constrained is used to select the applicable column of the associated table, and the following rules then apply:

a) for a type field, the component is constrained to be any value of any one of the types in any of the rows of that column;

b) for a value field, the component is constrained to be any one of the values in any of the rows of that column;

c) for a value set field, the component is constrained to be any one of the values in the value set in any one of the rows of that column.

NOTE – If, for some given object set, the above algorithms deliver no permissible value, then the constraint is always violated if that component is present in a value of a containing type.

Example

In the example in clause 10, case b) applies to the component **errorCategory**:

```
errorCategory    ERROR-CLASS.&category ({ErrorSet}) OPTIONAL,
```

with the associated table of **ErrorSet** restricting its possible values to **"A"** and **"B"**.

10.7 A component relation constraint is applied using the associated table of an information object set and the following production:

ComponentRelationConstraint ::=
"{" DefinedObjectSet "}" "{" AtNotation "," + "}"

AtNotation ::=
"@" ComponentIdList |
"@." Level ComponentIdList

Level ::= "." Level | empty

ComponentIdList ::= identifier "." +

10.8 Each "identifier" in the "ComponentIdList" identifies a component whose parent is a set, sequence or choice type, and shall be the last "identifier" if the component it identifies is not a set, sequence or choice type.

10.9 In the case where the parent is a set or sequence type the "identifier" shall be one of the "identifier"s of the "NamedType" in the "ComponentTypeLists" of that parent. In the case where the parent is a choice type the "identifier" shall be one of the "identifier"s of a "NamedType" alternative in the "AlternativeTypeLists" of that choice type.

10.10 The "AtNotation" provides a pointer to other components of the ASN.1 structure in which it appears. The parent structure for the first "identifier" in the "ComponentIdList" is determined as follows:

a) if the first alternative of "AtNotation" is selected (there is no "." following the "@"), then the parent structure is the outermost textually enclosing set type, sequence type or choice type.

b) if the second alternative is selected (there is a "." following the "@"), then the parent structure is obtained by moving upwards from the innermost textually enclosing set type or sequence type by a number of levels (set, set-of, sequence, sequence-of, choice) equal to the number of additional "." following "@.". The number of additional "." shall not exceed the number of constructions (set, set-of, sequence, sequence-of, choice) containing the innermost set or sequence type where the "AtNotation" occurs.

NOTE – The "AtNotation" is only permitted when it is textually within a set type or sequence type, and references some other field which is textually within the same set or sequence type, though possibly at a different level of nesting in constructions involving combination of sequence, sequence-of, set, set-of and choice types.

Example

In the following example "**@...**" illustrates case b) above:

```
ErrorMessage     ::=   SEQUENCE {
            severity         ERROR.&severity({Errors}),
            parameters  SEQUENCE OF SEQUENCE{
                  errorId    ERROR.&id({Errors}),
                  data       SEQUENCE OF SEQUENCE   {
                        value ERROR.&Type({Errors}{@severity,@...errorId}),
                        text  VisibleString}}}
```

10.11 The component where this notation is used is the referencing component, and the components identified by the "AtNotation"s are the referenced components.

10.12 The "ObjectSet" (see 10.3) or the "DefinedObjectSet" (see 10.7) is the constraining set, and the associated table derived from it (as specified in ITU-T Rec. X.681 | ISO/IEC 8824-2, clause 13) is the constraining table.

10.13 The component relation constraint can only be applied to an ASN.1 type which is textually within an enclosing "Type" (the constrained type) which textually contains all the referenced components. The constrained type is defined to be the innermost "Type" which satisfies the above condition.

Example

In the example in clause 10, the constrained type is **ErrorReturn**.

NOTE – In some respects it is possible to regard the application of this constraint as using the values of the referenced components to identify a row in the constraining table, and then using the value of the appropriate column to constrain the referencing component. With this view, the referenced components themselves could not be regarded as constrained.

However, the approach taken below is slightly different. It regards the constraint as operating over all possible values of the constrained type (which, as explained above, is not that of the referencing component), and selecting some of those values as satisfying the constraint. This approach makes it possible to discuss questions about values of the constrained type that do not contain values of either the referencing component or of one or more of the referenced components (because they were optional, or in choices), and values of the constrained type in which one of the referenced components has a value which does not correspond to any row in the constraining table.

10.14 The referencing and all the referenced components are required to be "ObjectClassFieldType"s referencing the same class. The constraining set is required to be an information object set of this class. The referenced components are required to be value fields or value set fields constrained by the same object set as the referencing component.

Example

In the example in clause 10, the "ObjectClassFieldType"s are all of class **ERROR-CLASS**, as is the constraining set, which is **ErrorSet**.

10.15 The following paragraphs determine the set of values of the constrained type that satisfy this constraint.

10.16 If the referencing component is absent in a value of the constrained type, that value always satisfies the constraint.

Example

In the example in clause 10, if the component **errors** is missing, then the constraints on **errors** are satisfied.

10.17 If any referenced component is absent in a value of the constrained type, that value does not satisfy the constraint unless the referencing component is also absent, in which case the constraint is always satisfied.

10.18 If all referenced components are present and the referencing component is present, then the constraint is not satisfied unless there exists in the constraining table one or more selected rows such that, for each selected row:

a) every referenced component which is a value field has a value that is the value of the corresponding column of the selected row;

b) every referenced component which is a value set field has a value that is one of the values in the value set of the corresponding column of the selected row.

10.19 The constraint is then satisfied if and only if the referencing component satisfies a simple table constraint (as defined above) obtained by applying a table containing only the selected rows to the referencing component.

Example

In the example in clause 10, the components **errorCategory**, **errorCode**, and **errorInfo** have to correspond to one of the rows of the associated table of **ErrorSet**.

10.20 If an "ObjectClassFieldType" is constrained by means of one or more "TableConstraint"s, and the "FieldName" denotes a type field, a variable-type value field, or a variable-type value set field, then in each instance of communication, the number of selected rows shall be exactly one if one of the referenced components is an identifier field, otherwise at least one shall be selected.

Example

In the example in clause 10, if there had been a further object **{"B" 2 PrintableString}**, then there could be more than one selected row.

11 Contents constraints

11.1 A contents constraint is specified by the syntax:

```
ContentsConstraint ::=
        CONTAINING Type
    |   ENCODED BY Value
    |   CONTAINING Type ENCODED BY Value
```

11.2 "Value" shall be a value of type object identifier.

11.3 The "ContentsConstraint" shall only be applied to octet string types and to bit string types defined without a "NamedBitList". Such constrained types shall not have further constraints applied to them, either directly or through the use of "typereference" names.

11.4 The first production of "ContentsConstraint" specifies that the abstract value of the octet string or bit string is the encoding of an (any) abstract value of "Type" that is produced by the encoding rules that are applied to the octet string or bit string. The following restrictions apply:

a) If this constraint is applied to an octet string, it is a specification error if any encoding of an abstract value of "Type" is not a multiple of eight bits.

b) If the octet string or bit string has a length constraint, the abstract values of "Type" are constrained to be those whose encoding can be contained within the constrained octet string or bit string. It is a specification error if there are no such abstract values.

11.5 The second production of "ContentsConstraint" specifies that the procedures identified by the object identifier value "Value" shall be used to produce and to interpret the contents of the bit string or octet string. If the bit string or octet string is already constrained, it is a specification error if these procedures do not produce encodings that satisfy the constraint.

11.6 The third production of "ContentsConstraint" specifies that the abstract value of the octet string or bit string is the encoding of an (any) abstract value of "Type" that is produced by the encoding rules identified by the object identifier value "Value". The following restrictions apply:

a) If this constraint is applied to an octet string, it is a specification error if any encoding of an abstract value of "Type" is not a multiple of eight bits.

b) If the octet string or bit string has a length constraint, the abstract values of "Type" are constrained to be those whose encoding can be contained within the constrained octet string or bit string. It is a specification error if there are no such abstract values.

Annex A

Constraining instance-of types

(This annex forms an integral part of this Recommendation | International Standard)

A.1 This annex specifies the application of constraints to "InstanceOfType" as defined in ITU-T Rec. X.681 | ISO/IEC 8824-2, Annex C.

A.2 The only constraint that can be applied to such a type is the simple table constraint, as specified in clause 10. The equivalent sequence type of the "InstanceOfType", when constrained in this way is:

```
SEQUENCE {
type-id   <DefinedObjectClass>.&id({  <DefinedObjectSet>  }),
value [0] <DefinedObjectClass>.&Type({  <DefinedObjectSet>  }{@.type-id})
}
```

where "<*DefinedObjectClass*>" is replaced by the particular "DefinedObjectClass" used in the "InstanceOfType" notation and "<*DefinedObjectSet*>" by the particular "DefinedObjectSet" used in the simple table constraint.

A.3 Where multiple constraints are applied to the instance-of type, each one produces a constraint of the above form, so that there are multiple constraints applied to each element of the equivalent sequence type.

A.4 Example

An example, building on the example in ITU-T Rec. X.681 | ISO/IEC 8824-2, C.10, is as follows.

The type:

```
INSTANCE OF MHS-BODY-CLASS ({PossibleBodyTypes})
```

has an equivalent sequence type of:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
type-id     MHS-BODY-CLASS.&id ({PossibleBodyTypes}),
value [0] MHS-BODY-CLASS.&Type ({PossibleBodyTypes}{@.type-id})
}
```

Here the **type-id** component of the sequence is constrained to take the value of the **&id** field of one of the **PossibleBodyTypes**, while the **value** component is constrained to be any value of the **&Type** field of that same information object.

In this case, the **PossibleBodyTypes** would likely be a parameter of the specification (see ITU-T Rec. X.683 | ISO/IEC 8824-4, clause 10 and A.8) which might not be resolved until a Protocol Implementation Conformance Statement (PICS) is produced, making the above constraints variable constraints as defined in ITU-T Rec. X.683 | ISO/IEC 8824-4, 10.3.