INTERNATIONAL STANDARD

ISO/IEC 8825-6

Fourth edition
2021-06

Information technology — ASN.1 encoding rules —

Part 6: Registration and application of PER encoding instructions

Technologies de l'information — Règles de codage ASN.1 —

Partie 6: Enregistrement et application des instructions de codage PER

Reference number
ISO/IEC 8825-6:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs)

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document 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 X.695 (02/2021).

This fourth edition cancels and replaces the third edition (ISO/IEC 8825-6:2015), which has been technically revised.

A list of all parts in the ISO/IEC 8825 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

CONTENTS

Page

Introduction

Rec. ITU-T X.680 | ISO/IEC 8824-1 makes syntactic provision for the application of encoding instructions to modify the behaviour of a particular set of encoding rules, identified by an encoding reference (see Rec. ITU-T X.680 | ISO/IEC 8824-1).

Rec. ITU-T X.691 | ISO/IEC 8825-2 specifies the BASIC-PER and CANONICAL-PER encoding rules, each with two variants: the ALIGNED variant and the UNALIGNED variant. The PER encoding instructions allow minor variations to be made in parts of the UNALIGNED variant of a BASIC-PER and CANONICAL-PER encoding. They have no effect on the ALIGNED variant of these encodings.

NOTE – The purpose of PER encoding instructions is to ease the task of producing an ASN.1 specification, which when encoded by the UNALIGNED variant of a PER encoding produces bit-patterns that exactly match those of a legacy protocol. It is unusual for the ALIGNED variant to be used for this purpose, and so for simplicity all PER encoding instructions have no effect on the ALIGNED variant.

This Recommendation | International Standard specifies the use of type prefixes and encoding control sections (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 31.3 and clause 54) to associate one or more PER encoding instructions with an ASN.1 type. Where an encoding instruction is associated with an ASN.1 type, specific clauses in Rec. ITU-T X.691 | ISO/IEC 8825-2 are amended according to the specification of the encoding instruction. These mechanisms are similar to those for the application of XER encoding instructions specified in Rec. ITU-T X.693 | ISO/IEC 8825-4.

This Recommendation | International Standard also specifies the procedures for the operation of a Registration Authority to receive, record and publish the specification of PER encoding instructions that are agreed from time to time. The Registration Authority is the ITU Telecommunication Standardization Bureau, and the form of publication is an Implementers' Guide for ASN.1. This Guide will be available freely on an ITU-T web-site.

This Recommendation | International Standard also specifies the procedures to be used for the approval of new PER encoding instructions. Broadly, these procedures involve the prior publication in the Implementers' Guide of a proposed new encoding instruction, with a later publication announcing that the new encoding instruction has been approved by a simple resolution of the relevant Study Group of ITU-T and the relevant Sub-Committee of ISO/IEC JTC 1.

Clauses 6 to 9 specify the operation of the Registration Authority for PER encoding instructions.

Clauses 10 to 13 specify the application of PER encoding instructions to an ASN.1 specification.

Annex A is informative and contains an example of the application of PER encoding instructions using encoding prefixes.

Annex B is informative and contains an example of the application of the same PER encoding instructions using an encoding control section.

Annex C is informative and summarizes the productions defined in this Recommendation | International Standard.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – ASN.1 encoding rules: Registration and application of PER encoding instructions

1 Scope

This Recommendation | International Standard:

a) specifies the information needed and the format to be used for specifying PER encoding instructions;

b) specifies the mechanisms for approving new PER encoding instructions from time to time and the operation of the Registration Authority for PER encoding instructions;

c) specifies the means of associating a PER encoding instruction with an ASN.1 type using both type prefixes and an encoding control section.

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

– Recommendation ITU-T X.680 (2021) | ISO/IEC 8824-1:2021, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

– Recommendation ITU-T X.691 (2021) | ISO/IEC 8825-2:2021, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.

NOTE – The references above shall be interpreted as references to the identified Recommendations | International Standards together with all their published amendments and technical corrigenda.

3 Definitions

For the purposes of this Recommendation | International Standard, the definitions of Rec. ITU-T X.680 | ISO/IEC 8824-1 apply. The following additional definitions apply.

3.1 associated encoding instructions (for a type): A set of PER encoding instructions associated with a type.

3.2 final encoding instructions (for a type): The set of PER encoding instructions associated with a type as a result of the complete ASN.1 specification, and which are applied in producing encodings of that type.

3.3 identifying keyword: A word or hyphenated word that identifies a PER encoding instruction.

3.4 inherited encoding instructions: PER encoding instructions that are associated with the type identified by a type reference.

3.5 Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1: A group established in accordance with Rec. ITU-T A.23, Annex A and ISO/IEC JTC 1 Directives Edition 5 Version 2.0, subclause 2.6.4 and Annex K, clause 8 to progress work on Joint Text in relation to Abstract Syntax Notation One (ASN.1).

3.6 PER encoding instructions (PER EIs): Notation used to change the unaligned PER encoding of a type (or of a component of a type).

NOTE – PER encoding instructions are included in either a PER type prefix (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 31.3) or a PER encoding control section (see Rec. ITU-T X.680 | ISO/IEC 8824-1, clause 54).

3.7 PER EI proposal: A proposal for a new PER encoding instruction that is progressing to either the REJECTED or the APPROVED state.

3.8 PER EI change proposal: A proposal for deletion of or change to an APPROVED PER encoding instruction that is progressing to either the REJECTED or the APPROVED state.

NOTE – PER EI change proposals are expected to be rare occurrences, and due consideration will be needed to backwards compatibility considerations.

3.9 **prefixed encoding instructions**: PER encoding instructions that are assigned using a type prefix (see Annex A and Rec. ITU-T X.680 | ISO/IEC 8824-1, 31.3).

NOTE – Prefixed encoding instructions can delete, replace, or add to the associated encoding instructions of a type.

3.10 **relevant Study Group**: The ITU-T Study Group that is responsible for the Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1.

3.11 **relevant Sub-Committee**: The ISO/IEC JTC1 Sub-Committee that is responsible for the Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1.

3.12 **targeted encoding instructions**: PER encoding instructions that are assigned using a target list in a PER encoding control section (see Annex B and Rec. ITU-T X.680 | ISO/IEC 8824-1, clause 54).

NOTE – Targeted encoding instructions can delete, replace, or add to the associated encoding instructions of a type.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

ASN.1	Abstract Syntax Notation One
ECN	Encoding Control Notation
EI	Encoding Instruction
PER	Packed Encoding Rules

5 Notation

5.1 This Recommendation | International Standard references and uses the notation defined by Rec. ITU-T X.680 | ISO/IEC 8824-1, clause 5 for the specification of the syntax of PER encoding instructions via a set of productions.

5.2 All lexical items used in these productions are defined in Rec. ITU-T X.680 | ISO/IEC 8824-1, clause 12.

5.3 In accordance with Rec. ITU-T X.680 | ISO/IEC 8824-1, 31.3.2, this Recommendation | International Standard specifies the "EncodingInstruction" production (see 11.2).

NOTE – The "**[**" and "**]**" lexical items never appear in an "EncodingInstruction" production.

5.4 In accordance with Rec. ITU-T X.680 | ISO/IEC 8824-1, 54.4, this Recommendation | International Standard specifies the "EncodingInstructionAssignmentList" production (see 12.1.2).

NOTE – The **END** and **ENCODING-CONTROL** lexical items never appear in the "EncodingInstructionAssignmentList" production.

6 Information to be provided to specify a PER encoding instruction

6.1 The specification of a PER encoding instruction shall consist of the following information:

a) a short informative title (such as "Use and size of a length field ");

b) the value for the identifying keyword, which shall be distinct from that used for any earlier PER EI;

c) an illustration of the syntax of the EI, enclosed in square brackets ("[" and "]");

d) a simple description of the EI and its purpose;

e) a full specification of the types to which this EI applies and its effect in each case;

f) a complete specification of an "EIDetail" production (see 11.4), with the semantics associated with each element;

g) either:

1) an amendment to specific text in Rec. ITU-T X.691 | ISO/IEC 8825-2 that shall be applied to the encoding of any type to which the EI is applicable (see d) above) and that has this EI in its set of final encoding instructions; or

NOTE 1 – If the EI is not applicable to the type, then the amendment is not applied in the encoding of that type.

2) an ECN specification of the effects of the encoding instruction; or

3) any other clear and implementable statement of the effects of the EI;

4) any combination of 1) to 3).

NOTE 2 – Whilst option 3) is clearly desirable, it may be necessary to progress with options 1) or 4) and add the others later so as not to delay approval of the EI.

NOTE 3 – The choice of options 1) to 4) has to be determined by whether the specification is sufficiently clear and precise to enable interworking implementations of encoders and decoders to be produced.

h) a statement that the clauses or subclauses of Rec. ITU-T X.691 | ISO/IEC 8825-2 that are being amended are disjoint from those amended by any previously approved encoding instruction that has an overlapping list of types to which they can be applied.

7 Status of a PER EI proposal during the approval process

7.1 A PER EI proposal can be in one of the following categories:

– **Considered but rejected (REJECTED)**: There has been some (documented) discussion, but this PER EI proposal is unlikely to be progressed at this time.

– **Suggested as possibly useful (POSSIBLE)**: The PER EI proposal is pending further discussion.

– **Under active development (NEEDED)**: There is agreement that a PER EI performing this function is needed, but the precise specification has still to be developed. Work is in progress to refine the specification of this PER EI.

– **Conditionally approved (READY)**: The full specification is available and ready for implementation. It has been approved by either the relevant Study Group or by the relevant Sub-Committee by a Resolution at a Plenary meeting.

– **Approved (APPROVED)**: The full specification of the PER EI has been published in the READY state for at least six months, and has been approved by both the relevant Study Group and the relevant Sub-Committee by Resolutions at Plenary meetings.

NOTE – PER EI proposals in the **NEEDED**, **READY**, and **APPROVED** categories are published by the Registration Authority (see clause 9). **REJECTED** and **POSSIBLE** PER EI proposals are recorded by the Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1 in standing documents.

7.2 A PER EI proposal will normally progress from an initial proposal into either the **REJECTED** or **POSSIBLE** or **NEEDED** category, and then to the **READY** and later to the **APPROVED** category. However, at any stage prior to moving to the **APPROVED** category, it can be moved to the **REJECTED** category.

NOTE – This will typically occur if problems are found with its implementation.

7.3 Once a PER EI proposal is in the **APPROVED** category (the PER EI is approved), it shall only be changed or removed by a PER EI change proposal that shall move through all the above stages with the same approval processes (specified in clause 8) before it results in the removal or modification of an **APPROVED** PER EI.

8 Approval process

8.1 Any member of the relevant Study Group and any member of the relevant Sub-Committee can generate a new PER EI proposal by submitting at least the information in 6.1 a) to d) in the format of Annex A. Additional free-form text with suggestions for e) to g) may also be included, and submitters of new EIs are encouraged to provide the full information required for 6.1.

8.2 Discussion of the proposal shall take place at the next available main or interim meeting of either the relevant Study Group or the relevant Sub-Committee. The proposal shall, after discussion, be placed in the category:

a) **REJECTED**: minutes of the meeting shall record the reasons for the rejection; or

b) **POSSIBLE**: details of the PER EI proposal shall be listed in a Standing Document of the Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1, recording proposed PER EIs, with the dates it was proposed, considered and details of the proposer and any relevant discussion; or

c) **NEEDED**: details of the PER EI proposal shall be published (see clause 9).

8.3 A PER EI proposal shall progress to the **NEEDED** category from any interim or main meeting of the relevant Study Group or the relevant Sub-Committee if and only if those present at the meeting agree unanimously and there is text available for the public announcement, with the information required for 6.1 a) to f) approved by the meeting. Details of the PER EI shall be published.

NOTE – This information is sufficient for syntax tools to be produced, and for users to include the PER EI in their draft specifications.

8.4 A PER EI proposal shall progress to the **READY** category from any main meeting of the relevant Study Group or the relevant Sub-Committee if and only if there is a formal Resolution at that meeting, based on availability and review of complete text for all of 6.1 a) to h), that the PER EI proposal is approved by that relevant Sub-Committee or relevant Study Group. The revised details of the EI shall be published.

8.5 A PER EI proposal shall progress to the **APPROVED** status if and only if the following conditions are satisfied:

a) the PER EI proposal has been in the **READY** category for at least six months;

b) both the relevant Study Group and the relevant Sub-Committee have passed a formal Resolution at a Plenary meeting that they approve the PER EI.

9 Publication by the Registration Authority

9.1 All Registration and publication shall be via a Web page on a web-site of the relevant Study Group maintained by the ITU Telecommunication Standardization Bureau that acts as the formal Registration Authority. The new contents of this page shall be provided by the Joint ITU-T | ISO/IEC JTC1 Collaborative Team for ASN.1 when the status of a PER EI proposal changes (see clause 8 for when such changes can occur).

NOTE 1 – The Web page at the time of publication can be located via the URL http://www.itu.int/ITU-T/studygroups/com17/, under the heading "Registration - Assignment", title "ASN.1 PER EIs".

NOTE 2 – Should these pages be moved, their new location can be obtained by an e-mail addressed to tsbmail@itu.int, asking to be put in contact with the ITU-T Rapporteur responsible for the maintenance of Rec. ITU-T X.695 | ISO/IEC 8825-6.

9.2 All PER EI proposals in the **NEEDED**, **READY** or **APPROVED** stage shall be published.

9.3 All PER EI change proposals in the **NEEDED** or **READY** stage shall be published.

9.4 A PER EI change proposal that moves to the **APPROVED** stage shall result in an archive record of the previous PER EI and an appropriate change to the modified PER EI, with an explanation of any backwards compatibility considerations.

10 Restrictions on the use of PER Encoding Instructions

10.1 The application of PER EIs can prevent some abstract values of the type from being encoded by PER. Depending on the intended application, this may or may not matter. It may be considered desirable, however, to additionally provide an explicit constraint on the type to ensure that all allowed values can be encoded by all encoding rules.

10.2 Where a PER EI implicitly forbids the encoding of some abstract values, the specification of the EI shall state that use of this EI restricts the abstract values that can be encoded.

EXAMPLE 1: Application of a PER EI to terminate an ASCII string with a NULL terminating character cannot be encoded by PER if the string contains a NULL character. The designer may choose to apply a normal constraint to restrict the string to non-NULL characters.

NOTE – This makes the specification more verbose, and arguably less clear, but ensures that relaying between different encoding rules is possible.

EXAMPLE 2: Application of a PER EI to use a 16 bit-field for the encoding of an INTEGER cannot be encoded by PER if the INTEGER is too large, and adding a constraint to a range that can be encoded in a 16-bit field should be considered.

10.3 A PER EI shall not be applied to a type that is extensible for PER encoding (see Rec. ITU-T X.691 | ISO/IEC 8825-2, 3.7.11).

NOTE – This applies to the type itself. Extensibility of a component does not restrict the application of PER EIs to other components or to the type itself.

11 Assigning a PER EI to an ASN.1 type using a type prefix

11.1 PER encoding instructions can be assigned to (or removed from) ASN.1 types (using either of the "EncodingInstruction" production alternatives) in a PER type prefix.

NOTE – The effect of multiple assignments of encoding instructions is specified in clause 13.

11.2 The PER "EncodingInstruction" production is:

```
EncodingInstruction ::=
        PositiveInstruction
        | NegatingInstruction

PositiveInstruction ::=
        IdentifyingKeyword
        EIDetail

NegatingInstruction ::=
        NOT PositiveInstruction
```

IdentifyingKeyword ::= encodingreference

11.3 The "IdentifyingKeyword" has the same syntax as an "encodingreference" (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 12.25), but has different semantics. Its value shall be determined for each PER EI, and shall be unambiguous within the set of all PER EIs. Its purpose is to identify a particular PER EI.

11.4 The "EIDetail" production shall be specified for each EI (see 6.1), and shall not contain the "⟦" and "⟧" lexical items.

11.5 An encoding instruction in a type prefix (or in an encoding control section – see 12.1.5) can be a positive instruction, used to add or to replace an encoding instruction (use of "PositiveInstruction"), or a negating instruction used to cancel one or more associated encoding instructions (use of "NegatingInstruction").

11.6 If the "Type" in a "TypeAssignment" (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 16.1) has final encoding instructions, all uses of the corresponding "typereference" (in the module containing the "TypeAssignment" or in some other module) inherit its final encoding instructions.

12 Assigning a PER encoding instruction using an encoding control section

12.1 The encoding instruction assignment list

12.1.1 PER encoding instructions can also be assigned to ASN.1 types in a PER encoding control section using the "EncodingInstructionAssignmentList" production.

12.1.2 The PER "EncodingInstructionAssignmentList" production is:

EncodingInstructionAssignmentList ::=
TargetedEncodingInstruction
EncodingInstructionAssignmentList ?

TargetedEncodingInstruction ::=
"[" EncodingInstruction "]"
TargetList

12.1.3 The "EncodingInstruction" production is defined in 11.2.

12.1.4 Each use of an "EncodingInstruction" in an encoding control section assigns that PER encoding instruction to the occurrences of "Type" that are identified in the "TargetList" of the "TargetedEncodingInstruction". The "TargetList" production and the targets it identifies are specified in 12.2.

12.1.5 Subclauses 11.5 and 11.6 also apply to encoding instructions in an encoding control section.

12.2 Identification of the targets for a PER encoding instruction using a target list

12.2.1 General rules

12.2.1.1 All targets are an occurrence of the "Type" production within the ASN.1 module.

NOTE – Multiple targets, in the same or in different ASN.1 type assignments, can be specified. A target that is the entire module, or all occurrences within the module of a built-in type or constructor can also be specified. Thus (using a PER encoding control section) a single "EncodingInstruction" can be used to assign a particular PER encoding instruction to all the types in an ASN.1 module that require that encoding instruction to be assigned.

12.2.1.2 In identifying the target(s) for the assignment of a PER encoding instruction, the production "TargetList" is used. This is defined in the following subclauses.

12.2.1.3 The "TargetList" production is:

TargetList ::=
Targets "," +
| empty

Targets ::=
TypeIdentification
| BuiltInTypeIdentification
| IdentifiersInContext

12.2.1.4 If the "TargetList" is a list of one or more "Targets" productions, then each of the "Targets" identifies one or more targets ("Type"s to which the encoding instruction is assigned).

12.2.1.5 The PER encoding instruction is assigned to all the types identified by the "TargetList" as specified in 12.2.1.9 to 12.2.1.14.

NOTE – It would be unusual, but not illegal, for a given "Type" to be identified more than once in the target list. In such cases, clause 13 applies.

12.2.1.6 (Tutorial) Identification of the target(s) (and possible qualifying information) by the "Targets" production uses one of four basic forms:

a) use of a "typereference" (see 12.2.2), possibly followed by a dot-separated list of identifiers (or an asterisk to denote the single component of a sequence-of or set-of), identifying either:
   1) the "Type" in a type assignment (if there is no dot-separated list of identifiers); or
   2) the "Type" in a component of a type definition (which can include top-level components introduced by the **COMPONENTS OF** construct – see 12.2.1.11);

b) use of **ALL** as the last identifier in the a) form, identifying all of the "Type"s textually present in the type definition (that is identified by the preceding type reference and dot-separated list of identifiers);

c) use of a "BuiltInTypeName" (see 12.2.3), identifying all "Type"s in the module that are defined by use of the corresponding built-in type name or constructor;

d) use of a list of "identifier"s followed by **IN** (or **ALL** followed by **IN,** or **COMPONENTS** followed by **IN)** and the a) form above (see also 12.2.4), identifying:
   1) the "Type" of the identified components of the a) form; or
   2) (use of **ALL**) all "Type"s that textually occur within the "Type" identified by the a) form; or
   3) (use of **COMPONENTS**) all "Type"s that are the top-level components of the "Type" identified by the a) form.

NOTE 1 – The term "type definition" used in a) and b) above emphasizes that only textually present identifiers can be used. Identifiers cannot be used if the "Type" is a type reference.

NOTE 2 – In general a component can be referenced by use of a) or d) above. If more than one component of a type is to be referenced, then d) would be preferred as it is less verbose, otherwise a) would be preferred. This is a matter of style.

12.2.1.7 A bitstring or octetstring type with a contents constraint that contains a type shall be treated as a type with a single component, using "*****" as the component identifier, for the purpose of assigning a targeted instruction to the "Type" in the contents constraint.

12.2.1.8 A type definition that is a sequence-of or a set-of shall be treated as a type with a single component, using "*****" as the component identifier, for the purpose of assigning a targeted instruction to the "Type" that is the component of the sequence-of or set-of.

NOTE – It is also possible to identify this single component using the component identifier (if present).

12.2.1.9 If a target is the use of a dummy parameter of a parameterized type, the target inherits the final encoding instructions of the actual parameter before encoding instructions targeting the dummy parameter are assigned. The specification is legal only if the resulting final encoding instructions for all instantiations of the parameterized type are legal.

NOTE 1 – If the parameterized type is exported, the final encoding instructions for its dummy parameters are carried with it.

NOTE 2 – There are no mechanisms provided to assign encoding instructions directly to the "Type" of an actual parameter in an instantiation of a parameterized type.

12.2.1.10 If the target is a "SelectionType", the target inherits the final encoding instructions of the selected alternative of the choice type referenced by the selection type, after which encoding instructions assigned to the "SelectionType" are assigned.

12.2.1.11 If the target is a component produced as a result of the **COMPONENTS OF** transformation, the target inherits the final encoding instructions of the component of the type referenced by the **COMPONENTS OF,** after which encoding instructions assigned to the components produced by the **COMPONENTS OF** are assigned. Any encoding instructions for the "Type" from which the components are extracted are ignored.

12.2.1.12 If the "Targets" production is "TypeIdentification", then the targets it identifies are specified in 12.2.2.

12.2.1.13 If the "Targets" production is "BuiltInTypeIdentification", then the targets it identifies are specified in 12.2.3.

12.2.1.14 If the "Targets" production is "IdentifiersInContext", then the targets it identifies are specified in 12.2.4.

EXAMPLE: The example below shows an ASN.1 type definition followed by two different ways of assigning PER encoding instructions in an encoding control section, and finally, the same ASN.1 type definition with the PER encoding instructions assigned using type prefixes. All three approaches result in the same encoding.

NOTE – A more extensive example is given in Annexes A and B.

The type definition is:

```
My-Type ::= SEQUENCE {
              field1  INTEGER,
              field2  CHOICE {
                         first   SEQUENCE OF INTEGER,
                         second  SEQUENCE OF OBJECT IDENTIFIER } }
```

PER encoding instructions to apply EI1 and EI2 in the encoding control section could be:

```
      [EI1] field1 IN My-Type
              [EI2] first IN My-Type.field2
```

Alternatively, they could be:

```
      [EI1] My-Type.field1
              [EI2] My-Type.field2.first
```

The type definition with type prefixes is:

```
My-Type ::= SEQUENCE {
              field1  [EI1] INTEGER,
              field2  CHOICE {
                         first   [EI2] SEQUENCE OF INTEGER,
                         second  SEQUENCE OF OBJECT IDENTIFIER } }
```

12.2.2 Target identification using an ASN.1 type reference and identifiers

12.2.2.1 The "TypeIdentification" production is:

TypeIdentification ::=
ALL
| typereference ComponentReference ?

ComponentReference ::=
"."
ComponentIdList

ComponentIdList ::=
ComponentId "." +

ComponentId ::=
identifier
| "*"
| ALL

12.2.2.2 A "TypeIdentification" of **ALL** identifies all "Type"s in "TypeAssignment"s in the module.

12.2.2.3 The "typereference" (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 12.2) shall be a type reference that is defined in the module.

12.2.2.4 A symbol **"*"** identifies the "Type" of the (sole) component of a sequence-of or set-of type, or the type in a contents constraint that contains a "Type".

NOTE – This form can be used even if the sequence-of or set-of component has an identifier, but the use of the "identifier" (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 12.3) should be preferred.

12.2.2.5 If **ALL** is used as a "ComponentId", it shall be the last "ComponentId" in the "ComponentIdList".

12.2.2.6 If the first "ComponentId" in the "ComponentIdList" (if present) is an identifier that is textually present (or results from use of **COMPONENTS OF**) as a component identifier in the "Type" identified by the "typereference", then it identifies the "Type" of that component. If it is not an identifier that is textually present (or results from use of **COMPONENTS OF**) as a component identifier in the "Type" identified by the "typereference", then this occurrence of "TypeIdentification" is not illegal, but does not identify any target.

NOTE – This requires that the type referenced by the "typereference" be a sequence, set, choice, sequence-of or set-of type definition, or a bitstring or an octetstring type definition with a contents constraint that contains a "Type".

12.2.2.7 If a subsequent "ComponentId" (except the last) in the "ComponentIdList" (if present) is an identifier that is textually present as a component identifier in the "Type" identified by the previous "ComponentId", then it identifies the "Type" of that component. If it is not a component identifier that is textually present in the "Type" identified by the previous "ComponentId", then this occurrence of "TypeIdentification" is not illegal, but does not identify any target.

NOTE – The first use of "ComponentId" can refer to components introduced by a **COMPONENTS OF**. Components of those components cannot be identified by subsequent "ComponentId"s.

12.2.2.8 If the last "ComponentId" in the "ComponentIdList" (if present) is:

a) an identifier that is textually present as a component identifier in the "Type" identified by the previous "ComponentId", then it identifies the "Type" of that component and the encoding instruction shall be assigned to that "Type"; or

b) the keyword **ALL**, then the encoding instruction shall be assigned to all "Type"s that are textually present in the type definition identified by the previous "ComponentId", which shall be a type with one or more components.

12.2.3 Target identification using a built-in type name

12.2.3.1 The "BuiltInTypeIdentification" production is:

```
BuiltInTypeIdentification ::=
    BIT STRING
    | BOOLEAN
    | CHARACTER STRING
    | CHOICE
    | DATE
    | DATE-TIME
    | DURATION
    | EMBEDDED PDV
    | ENUMERATED
    | EXTERNAL
    | GeneralizedTime
    | INSTANCE OF
    | INTEGER
    | NULL
    | ObjectDescriptor
    | OBJECT IDENTIFIER
    | OCTET STRING
    | REAL
    | RELATIVE-OID
    | SEQUENCE
    | SEQUENCE OF
    | SET
    | SET OF
    | TIME
    | TIME-OF-DAY
    | UTCTime
    | RestrictedCharacterStringType
```

12.2.3.2 The "BuiltInTypeIdentification" production specifies that the encoding instruction is to be applied to all textual occurrences within the module of the corresponding built-in type or of a type defined using the corresponding constructor.

12.2.3.3 The "RestrictedCharacterStringType" is defined in Rec. ITU-T X.680 | ISO/IEC 8824-1, clause 41.

12.2.4 Use of identifiers in context

12.2.4.1 The "IdentifiersInContext" production is:

```
IdentifiersInContext ::=
        IdentifierList
        IN
        TypeIdentification

IdentifierList ::=
        identifier "," +
    | ALL
        | COMPONENTS
```

12.2.4.2 "TypeIdentification" is defined in 12.2.2, and identifies a type defined in a type assignment statement in the module, or a component or sub-component of a type defined in the module.

12.2.4.3 The "Type" identified by the "TypeIdentification" shall be a sequence, set or choice type, and is called for the purposes of this clause the identified "Type".

NOTE – The "TypeIdentification" in "IdentifiersInContext" cannot be used for a sequence-of or set-of type. Such use is prohibited for clarity, as it would be no less verbose than direct use of "TypeIdentification" in "Targets".

12.2.4.4 Each "identifier" (see Rec. ITU-T X.680 | ISO/IEC 8824-1, 12.3) in "IdentifierList" shall be the "identifier" of a component of the identified "Type". The PER encoding instruction is assigned to the "Type" of all the components of the identified "Type" that have a component "identifier" in the "IdentifierList".

12.2.4.5 The use of **ALL** for "IdentifierList" specifies that all textually present components (and all textually present components of those components, to any depth) in the identified "Type" are targets to which the PER encoding instruction is being assigned.

12.2.4.6 The use of **COMPONENTS** for "IdentifierList" specifies that all components (at the first level) of the identified "Type" are targets to which the PER encoding instruction is being assigned.

13 Multiple assignment of PER encoding instructions

13.1 Order in which multiple assignments are considered

13.1.1 A "Type" which is not a "typereference" has initially an empty set of associated encoding instructions.

13.1.2 A "Type" which is a "typereference" (which may be imported) has initially the set of final encoding instructions of the "Type" which was assigned to it when it was defined.

13.1.3 Targeted encoding instructions for a "Type" (using an encoding control section) are assigned next, in the order in which the targeted encoding instructions appear in the encoding control section. If the "Type" is identified by more than one element of a "TargetList" (see 12.2), then that shall be treated as multiple assignments of the same encoding instruction to that "Type", in the order in which the elements occur in the "TargetList".

NOTE – The effect of 13.1.2 and 13.1.3 means that targeted assignment to a "Type" in a "TypeAssignment" is always over-ridden by a targeted assignment to a "Type" defined using the corresponding "typereference", no matter which targeted assignment appears first in the encoding control section. However, if a targeted assignment is made to all the components of a type, and also to an individual component of that type, the effect will depend on the order of the encoding instructions in the encoding control section.

13.1.4 Prefixed encoding instructions (using a type prefix) assigned to a type are considered next, with the rightmost (the innermost) prefixed encoding instruction considered first, and the leftmost (the outermost) prefixed encoding instruction considered last.

13.1.5 As specified in 12.2.1.9, encoding instructions are assigned to a dummy parameter only after the final encoding instructions for the actual parameter have been determined.

13.1.6 As specified in 12.2.1.10 and 12.2.1.11, a "SelectionType" and the components produced by a **COMPONENTS OF** transformation inherit first the final encoding instructions of the original type, and then have encoding instructions targeted at them applied.

13.1.7 Each assignment of an encoding instruction produces a new set of associated encoding instructions, as specified in 13.2 to 13.3.

13.2 Effect of assigning a negating encoding instruction

All assignments of a negating encoding instruction result in the removal (from the set of associated encoding instructions) of all preceding encoding instructions, and the set becomes empty.

NOTE – A negating encoding instruction never becomes part of the set of associated encoding instructions.

13.3 Multiple assignment of PER encoding instructions

NOTE – Multiple assignment of PER encoding instructions is expected to be rare, except where an encoding instruction is assigned globally, and an overriding (possibly negating) encoding instruction is assigned to specific types or components. This subclause specifies the rules if multiple assignment of PER encoding instructions occurs.

13.3.1 Assignments of positive encoding instructions result in the addition (to the set of associated encoding instructions) of that PER encoding instruction if there are no other associated encoding instructions with the same "IdentifyingKeyword".

13.3.2 If there is an encoding instruction with the same "IdentifyingKeyword" in the set of associated encoding instructions, then that encoding instruction is removed from the set, and the assigned PER encoding instruction is added.

NOTE – If encoding instructions are being assigned globally in an encoding control section, with the intention of overriding them in specific cases, then the overriding has to be done using either a type prefix or a later encoding instruction in the encoding control section, not an earlier one.

13.3.3 If a type appears in a "ContentsConstraint" or in a "TypeConstraint", then the final encoding instructions (as determined by the above rules) are used in determining the encoding of that type. If a type appears in any other ASN.1 constraint, then all associated encoding instructions are discarded.

Annex A

Example of the application of PER EIs using prefixed encoding instructions

(This annex does not form an integral part of this Recommendation | International Standard.)

A.1 This annex contains an example of the use of prefixed PER EIs to produce a variation of the PER encodings for the types in a module.

A.2 The example is based on an early text of part of an ISO/IEC Standard. The final version of the Standard differs significantly in the abstract syntax specified, so the example should be used solely for the purposes of illustration in this Recommendation | International Standard.

A.3 For clarity, the PER encoding instructions (and comments related to them) are shown in grey.

A.4 The module specified using prefixed PER EIs is:

```
SignatureSignRecordFormatModule
{iso standard 19794 signature-sign(7) modules(0) record-format(0) version(0)}
DEFINITIONS
PER INSTRUCTIONS
-- This specifies that PER Encoding Instructions are to be applied
AUTOMATIC TAGS ::=
BEGIN
    SignatureSignBlock ::= SEQUENCE {
            header      Header,
            body        Body}
    Header ::= SEQUENCE {
                formatId [NULL]
                     -- This specifies that the IA5String is to
                     -- be followed by a zero(NULL) octet in
                     -- the encoding.
                                  IA5String ("SDI"),
                standardVersion [NULL]
                                -- As above.
                                  IA5String (SIZE (3))
                     -- " 10" (space-one-zero) for this version --,
                channelInclusions       ChannelInclusions,
                channelDescriptions     ChannelDescriptions}
ChannelInclusions ::= SEQUENCE {
                x-included  BOOLEAN,
                y-included  BOOLEAN,
                z-included  BOOLEAN,
                vX-included BOOLEAN,
                vY-included BOOLEAN,
                aX-included BOOLEAN,
                aY-included BOOLEAN,
                t-included  BOOLEAN,
                dt-included BOOLEAN,
                f-included  BOOLEAN,
                s-included  BOOLEAN,
                tX-included BOOLEAN,
                tY-included BOOLEAN,
                az-included BOOLEAN,
                el-included BOOLEAN,
                r-included  BOOLEAN}
                        (WITH COMPONENTS
                          {x-included (TRUE),
                           y-included (TRUE)})
ChannelDescriptions ::= [OPTIONALITY-IN Header.channel-inclusions]
                          -- This specifies that the optionality bit-map is
                          -- taken from the channel inclusions.  The
                          -- channel-inclusions structure is needed because
                          -- the same bit-map controls the optionality
                          -- in each SamplePoint SEQUENCE in the
                          -- SamplePoints SEQUENCE OF.  It is also
                          -- desirable to make the bit-map
                          -- application-visible
            SEQUENCE {
                x       SignedChannelDescr      OPTIONAL,
                y       SignedChannelDescr      OPTIONAL,
                z       UnsignedChannelDescr    OPTIONAL,
```

```
                vX      SignedChannelDescr      OPTIONAL,
                vY      SignedChannelDescr      OPTIONAL,
                aX      SignedChannelDescr      OPTIONAL,
                aY      SignedChannelDescr      OPTIONAL,
                t       UnsignedChannelDescr    OPTIONAL,
                dt      UnsignedChannelDescr    OPTIONAL,
                f       UnsignedChannelDescr    OPTIONAL,
                s       UnsignedChannelDescr    OPTIONAL,
                tX      SignedChannelDescr      OPTIONAL,
                tY      SignedChannelDescr      OPTIONAL,
                az      UnsignedChannelDescr    OPTIONAL,
                el      UnsignedChannelDescr    OPTIONAL,
                r       UnsignedChannelDescr    OPTIONAL}
                  (CONSTRAINED BY {ChannelInclusions
                       -- Each element can be present if and only if permitted by
                       -- the ChannelInclusions -- })
    SignedChannelDescr ::= SEQUENCE {
                reserved      INTEGER (0..8),
                scalingValue  ScalingValue      OPTIONAL,
                min           SignedInt16       OPTIONAL,
                max           SignedInt16       OPTIONAL,
                mean          SignedInt16       OPTIONAL,
                std           UnsignedInt16     OPTIONAL}
    UnsignedChannelDescr ::= SEQUENCE {
                reserved            INTEGER (0..8),
                scalingValue        ScalingValue      OPTIONAL,
                min                 UnsignedInt16     OPTIONAL,
                max                 UnsignedInt16     OPTIONAL,
                mean                UnsignedInt16     OPTIONAL,
                std                 UnsignedInt16     OPTIONAL}
    ScalingValue ::= SEQUENCE {
                exponent [ENCODE-DIRECTLY] INTEGER (-16..15),
                             -- This ensures a two's complement
                             -- encoding, not an encoding from
                             -- the base of -16.
                fraction  INTEGER (0..2047)}
    Body ::= [SIZE 8] SEQUENCE {
              -- 8 bit optionality bit-map, with only one bit used.  Other
              -- bits will be set to zero by encoders, ignored by decoders.
                samplePoints [LENGTH 3][COUNT-OCTETS] SEQUENCE
                             -- Prevents optimisation for short
                             -- iterations, forcing 3 octets in
                             -- all cases
                             SIZE  (0..16777215) OF SamplePoint,
                extendedData [TERMINATED-BY-CARRIER]
                             -- The end of this octet string can only
                             -- be determined by running out of the
                             -- input buffer provided by the carrier
                             -- protocol.  It is a feature of this BDB
                             -- format that it relies on the CBEFF
                             -- carrier to delimit it
                             OCTET STRING OPTIONAL}
    SamplePoint ::=
                [OPTIONALITY-IN Header.channelInclusions]
                 -- As above
                SEQUENCE {
                x       SignedInt16       OPTIONAL,
                y       SignedInt16       OPTIONAL,
                z       UnsignedInt16     OPTIONAL,
                vX      SignedInt16       OPTIONAL,
                vY      SignedInt16       OPTIONAL,
                aX      SignedInt16       OPTIONAL,
                aY      SignedInt16       OPTIONAL,
                t       UnsignedInt16     OPTIONAL,
                dt      UnsignedInt16     OPTIONAL,
                f       UnsignedInt16     OPTIONAL,
                s       UnsignedInt8      OPTIONAL,
                tX      SignedInt16       OPTIONAL,
                tY      SignedInt16       OPTIONAL,
                az      UnsignedInt16     OPTIONAL,
                el      UnsignedInt16     OPTIONAL,
```

```
                r       UnsignedInt16    OPTIONAL}
            (CONSTRAINED BY {ChannelInclusions
                        -- Each element can be present if and only if
                        -- permitted by the channel inclusions -- })
UnsignedInt16 ::=     INTEGER (0..65535)
SignedInt16   ::=     [ENCODE-DIRECTLY] INTEGER (-32768..32767)
                            -- As above
UnsignedInt8  ::=     INTEGER (0..255)
END
```

Annex B

Example of the application of PER EIs using targeted encoding instructions

(This annex does not form an integral part of this Recommendation | International Standard.)

B.1 This annex contains an example of the use of targeted PER EIs to produce a variation of the PER encodings for the types in a module.

B.2 The example is the same as that used in Annex A and produces exactly the same encoding. Its advantage (over Annex A) is that it leaves the abstract syntax definition uncluttered, and is a small addition (textually) to the main specification. It also makes it clear that other encoding rules (and in-core data structures produced by a compiler) will allow the encoding of the full range of abstract values of the type. Its disadvantage is that it is less readable by a novice reader than Annex A. Protocol designers can use either form – or a mixture (not recommended!).

B.3 The module specified using targeted PER EIs is:

```
SignatureSignRecordFormatModule
{iso standard 19794 signature-sign(7) modules(0) record-format(0) version(0)}
DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN
    SignatureSignBlock ::= SEQUENCE {
             header      Header,
             body        Body}
    Header ::= SEQUENCE {
                  formatId               IA5String ("SDI"),
                  standardVersion        IA5String (SIZE (3))
                        -- " 10" (space-one-zero) for this version --,
                  channelInclusions      ChannelInclusions,
                  channelDescriptions    ChannelDescriptions}
    ChannelInclusions ::= SEQUENCE {
                  x-included  BOOLEAN,
                  y-included  BOOLEAN,
                  z-included  BOOLEAN,
                  vX-included BOOLEAN,
                  vY-included BOOLEAN,
                  aX-included BOOLEAN,
                  aY-included BOOLEAN,
                  t-included  BOOLEAN,
                  dt-included BOOLEAN,
                  f-included  BOOLEAN,
                  s-included  BOOLEAN,
                  tX-included BOOLEAN,
                  tY-included BOOLEAN,
                  az-included BOOLEAN,
                  el-included BOOLEAN,
                  r-included  BOOLEAN}
                        (WITH COMPONENTS
                          {x-included (TRUE),
                           y-included (TRUE)})
    ChannelDescriptions ::=
              SEQUENCE {
                  x       SignedChannelDescr      OPTIONAL,
                  y       SignedChannelDescr      OPTIONAL,
                  z       UnsignedChannelDescr    OPTIONAL,
                  vX      SignedChannelDescr      OPTIONAL,
                  vY      SignedChannelDescr      OPTIONAL,
                  aX      SignedChannelDescr      OPTIONAL,
                  aY      SignedChannelDescr      OPTIONAL,
                  t       UnsignedChannelDescr    OPTIONAL,
                  dt      UnsignedChannelDescr    OPTIONAL,
                  f       UnsignedChannelDescr    OPTIONAL,
                  s       UnsignedChannelDescr    OPTIONAL,
                  tX      SignedChannelDescr      OPTIONAL,
                  tY      SignedChannelDescr      OPTIONAL,
                  az      UnsignedChannelDescr    OPTIONAL,
                  el      UnsignedChannelDescr    OPTIONAL,
                  r       UnsignedChannelDescr    OPTIONAL}
                    (CONSTRAINED BY {ChannelInclusions
                    -- Each element can be present if and only if permitted by
                    -- the ChannelInclusions -- })
```