INTERNATIONAL STANDARD

ISO/IEC 10165-4

First edition
1992-09-01

AMENDMENT 2
1998-12-15

Information technology — Open Systems Interconnection — Structure of management information: Guidelines for the definition of managed objects

AMENDMENT 2: Addition of the NO-MODIFY syntax element and guideline extensions

Technologies de l'information — Interconnexion de systèmes ouverts — Structures des informations de gestion: Principes directeurs pour la définition des objets gérés

AMENDEMENT 2: Addition de l'élément de syntaxe «NO-MODIFY» et des extensions de principe directeur

Reference number
ISO/IEC 10165-4:1992/Amd.2:1998(E)

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. 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.

Amendment 2 to ISO/IEC 10165-4:1992 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 33, *Distributed application services*, in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.722/Amd.2.

Introduction

This Amendment adds the NO-MODIFY syntax element to the attribute properties of the package template in order to clearly specify that an attribute cannot be modified in subclasses and compatible classes of the managed object class.

This Amendment proposes that the **use of ASN.1:1994 should not be forbidden** (i.e. normative specification of ASN.1:1994 may be employed) in new and developing OSI Systems Management standards for the following reasons:

- ASN.1:1990 bugs/defects;
- ASN.1:1994 enhancements.

ASN.1:1990 and ASN.1:1994 modules can be mixed and are completely compatible using the guidelines in A.2 of Rec. ITU-T X.680 | ISO/IEC 8824-1, ASN.1 Specification of Basic Notation. This "mixed mode" use of ASN.1 places certain restrictions on allowed productions of both ASN.1:1990 and ASN.1:1994. If these guidelines are maintained, not only are ASN.1:1990 and ASN.1:1994 completely compatible, they are virtually indistinguishable and have identical encoding. Using these guidelines, new and developing OSI Systems Management standards may use both ASN.1:1990 and ASN.1:1994 for normative specification of syntax.

This Amendment introduces conventions for the specification of ASN.1 and GDMO directives in order to clearly identify specification and user options associated with ASN.1 modules and GDMO templates.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – STRUCTURE OF MANAGEMENT INFORMATION: GUIDELINES FOR THE DEFINITION OF MANAGED OBJECTS

AMENDMENT 2
Addition of the NO-MODIFY syntax element and guideline extensions

1) Subclause 2.1

Insert the following references by numerical order:

- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.690 (1994) | ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*
- ITU-T Recommendation X.691 (1995) | ISO/IEC 8825-2:1995, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER).*

2) Subclause 8.2

Add the following to the end of item m):

(Guidelines for the Production of Equivalent ASN.1:1990 and ASN.1:1994 Modules are given in clause 9.)

3) Subclause 8.4.2

Add the following to this subclause, following the `[SET-BY-CREATE]` *property in the* `propertylist`*. (The* `[SET-BY-CREATE]` *property was added in by Amendment 1):*

```
[NO-MODIFY]
```

4) Subclause 8.4.3.2

Add the following two new paragraphs before the last paragraph:

The absence of the `REPLACE` property may be used to specify that an attribute cannot be replaced for instances of a class but this absence does not preclude subclasses adding the `REPLACE` property. The `NO-MODIFY` property is present to explicitly specify that an attribute cannot be modified (is read-only) in the class having this property and in all

subclasses and in all compatible managed objects (i.e. managed objects behaving allomorphically to the class). This property is inconsistent with and shall not be present in a managed object class definition that has any of the REPLACE, GET-REPLACE, ADD, REMOVE, or ADD-REMOVE properties on the same attribute.

NOTE 3 – The NO-MODIFY property is not necessarily inconsistent with the REPLACE-WITH-DEFAULT property because this operation is often used with a meaning of "reset" that can be consistent with a manager's inability to control the attribute's value.

NOTE 4 – Before the NO-MODIFY property was added to GDMO, the convention was to specify that property in BEHAVIOUR templates or in documents referenced in BEHAVIOUR templates.

If it is desired that part of the definition of an attribute is that the attribute shall not be replaced in any class that specifies the attribute, then this constraint shall be specified in a BEHAVIOUR template referenced by the ATTRIBUTE template.

5) Index

Add new index entry:

NO-MODIFY 26-27

Change these index entries:

BEHAVIOUR 25-26, 30-32, 34-35, 37-40

REPLACE 14, 26-27

6) New clause 9

Add a new clause 9, following 8.11.3.5:

9 Guidelines for production of equivalent ASN.1:1994 and ASN.1:1990 modules

It is possible in developing standards, that normative ASN.1:1994 modules be provided. In order to allow the use of ASN.1:1994, it is recommended that an equivalent normative ASN.1:1990 module be provided subject to the following:

1) it has the same object identifier as the ASN.1:1994 module;

2) it is normative but the standard states that in the event of discrepancies between the ASN.1:1990 and ASN.1:1994 modules, the ASN.1:1994 module take precedence;

3) the standard states that use of ASN.1:1990 will be retained as long as needed.

NOTE 1 – ISO/IEC JTC 1/SC 21 rules dictate a periodic review for renewal of the ASN.1:1990 International Standards every one (1) year.[1] National Bodies are asked to consider the above when reviewing the ASN.1:1990 standards. This ensures that the ASN.1:1990 standards are retained as long as needed.

Also to reduce errors, it is recommended that the ASN.1:1990 module be a machine generated transformation of the ASN.1:1994 module since this transformation can be easily automated.

NOTE 2 – If an editor desires to use a commercial tool (e.g. made available to the standards community at virtually no cost, vendor XXX's ASN.1 Tools) for converting ASN.1:1994 to ASN.1:1990 to reduce the possibility of errors, it has been suggested that the editor should add a comment at the top of the generated code that says something like:

-- *XXX ASN.1 Tools used for conversion* --
-- *from ASN.1:1994 to ASN.1:1990* --

with the following Note:

NOTE – Although ISO cannot advertise that one software tool must be used as opposed to another, at present, XXX ASN.1 Tools is one of the software tools allowing the conversion from ASN.1:1994 to ASN.1:1990.

It should be noted that problems can be avoided if only the common subset of ASN.1:1990 and ASN.1:1994 is used. In this case, only the ASN.1:1994 module needs to be included in this standard.

[1] ISO/IEC JTC 1/SC 21 (SC21) reaffirmed the continuation of availability of the ASN.1:1990 standards for reasons of conformance and interpretability (in SC21 N 9001 rev). SC21 requested its WGs to continue to maintain these standards. An SC21 resolution to continue maintenance will be conducted at each SC21 meeting (currently, once a year).

9.1 Guidelines

The following guidelines must be followed:

1) A single systems management document may reference ASN.1 modules from both ASN.1:1990 and ASN.1:1994. However, any given module is required to completely conform to ***either*** ASN.1:1990 ***or*** ASN.1:1994, where directives defined in clause 10 are used to identify which version of the notation is being used in a particular module.

2) Type and value references may be imported in an ASN.1:1994 module from an ASN.1:1990 module as long as:

   a) ASN.1:1990 **MACROs** are not imported in an ASN.1:1994 module; therefore one cannot create an instance of a **MACRO** in an ASN.1:1994 module.

   b) Identifiers for **SET** and **SEQUENCE** and **CHOICE** values are present.

3) Also, type and value references may be imported in an ASN.1:1990 module from an ASN.1:1994 module as long as:

   – ASN.1:1994 **CHARACTER STRING, BMPStrings, UniversalStrings, EMBEDDED PDV** types are not imported. Since there is no ASN.1:1990 equivalent for these ASN.1:1994 types, their use is discouraged in ASN.1:1994 modules which require an equivalent ASN.1:1990 module. For this same reason, the use of the ASN.1:1994 **Tuple** type is forbidden in ASN.1:1994 modules which require an equivalent ASN.1:1990 module[2)].

   NOTE 1 – If the proposed guidelines are maintained, there are no adverse implications of importing type and value references from one version of ASN.1 to another because equivalent constructs exist in all cases.

4) The following ASN.1:1994 module shall be used for definitions of ASN.1 information object classes that are imported into ASN.1:1994 modules:

```
-- <ASN1.Version 1994 SMModule {joint-iso-itu-t ms(9) smi(1) part4(4)
--              asn1Module(2) 2} >--

SMModule {joint-iso-itu-t ms(9) smi(1) part4(4) asn1Module(2) 2}

DEFINITIONS ::= BEGIN

REGISTERED-AS ::= TYPE-IDENTIFIER

-- TYPE-IDENTIFIER is defined in ISO/IEC 8824-1 and is available in any module
-- without the necessity for importing it and is defined as:
-- TYPE-IDENTIFIER::= CLASS
-- {
-- &id OBJECT IDENTIFIER UNIQUE,
-- &Type
-- }
-- WITH SYNTAX {&Type IDENTIFIED BY &id}

INFO-REPLY-IDENTIFIER ::= CLASS
        {
            &Info OPTIONAL,
            &Reply OPTIONAL,
            &registeredAs OBJECT IDENTIFIER UNIQUE
        }

WITH SYNTAX {INFO &Info REPLY &Reply IDENTIFIED BY &registeredAs}

RegisteredAsTable REGISTERED-AS ::= {...}

InfoReplyTable INFO-REPLY-IDENTIFIER ::= {...}

-- RegisteredAsTable to be filled in by GDMO ATTRIBUTE and PARAMETER
-- Templates
-- InfoReplyTable to be filled in by GDMO ACTION and NOTIFICATION Templates

END
```


---

2) **Tuple** is the name of the ASN.1:1994 production that allows control characters to be inserted in the value notation of an IA5String, something that cannot be done in ASN.1:1990. For example, ... **greetings IA5String ::= {"hello", cr, "there"}** inserts a carriage return between **"hello"** and **"there"** (**cr** is imported from a module defined in ITU-T Rec. X.680 | ISO/IEC 8824-1 and is equated to a literal carriage return).

5) Editors define ASN.1:1994 modules as in the following example:

```
-- <ASN1.Version 1994 ExampleModule > --

ExampleModule {-- a valid object identifier goes here --}

DEFINITIONS ::= BEGIN

IMPORTS
REGISTERED-AS,
INFO-REPLY-IDENTIFIER,
RegisteredAsTable,
InfoReplyTable
FROM SMModule {joint-iso-itu-t ms(9) smi(1) part4(4) asn1Module(2) 2};

Foo::=      SEQUENCE {
            id1 REGISTERED-AS.&id ({RegisteredAsTable}),
            syntax1 REGISTERED-AS.&Type ({RegisteredAsTable} {@.id1})}

Bar::=      SEQUENCE {
            id2 REGISTERED-AS.&id ({RegisteredAsTable}),
            syntax2 SEQUENCE OF REGISTERED-AS.&Type ({RegisteredAsTable} {@id2})}

firstExtensionId OBJECT IDENTIFIER ::= {1 3 17 103 10 1}

FirstExtensionInfo ::= PrintableString

-- Illustrates use of a contained subtype constraint of an open type --

FooBar ::=  Foo (WITH COMPONENTS {
            id1 (firstExtensionId),
            syntax1(FirstExtensionInfo)} )
END
```

Note that since GDMO is being used in combination with the ASN.1 **REGISTERED-AS** information object class, **FooBar** is a duplication of information. That is, the GDMO specification plus **REGISTERED-AS** in ASN.1 is equivalent to the inner type constraint on **Foo. FooBar** simply illustrates that while an open type can be constrained to any type, an **ANY/ANY DEFINED BY** cannot be constrained to anything other than an **ANY/ANY DEFINED BY** type in ASN.1:1990 and it illustrates how to do a mapping from an open type so constrained in ASN.1:1994 to an ASN.1 comment in ASN.1:1990.

6) Editors convert ASN.1:1994 module using the following instructions to produce an equivalent ASN.1:1990 module:

a) Remove the portion of the **IMPORTS** statement that refers to module SMModule. This gets rid of the imported definitions of the information object classes **REGISTERED-AS** and **INFO-REPLY-IDENTIFIER**.

b) Convert all open-type references to **ANY** or **ANY DEFINED BY** types. To do so, convert all ASN.1 syntax of the form according to the following:

First, convert:

FROM	TO
REGISTERED-AS.&id	**OBJECT IDENTIFIER**

If "**REGISTERED-AS.&Type**" is a component of a **SET** or **SEQUENCE**

and it is defined in the same **SET** or **SEQUENCE** as "**id**"

then convert:

FROM	TO
REGISTERED-AS.&Type ({RegisteredAsTable} {@id1})	**ANY DEFINED BY id**
REGISTERED-AS.&Type ({RegisteredAsTable} {@.id1})	**ANY DEFINED BY id**

else convert:

FROM	TO
REGISTERED-AS.&Type ({RegisteredAsTable} {@id1})	**ANY**
REGISTERED-AS.&Type ({RegisteredAsTable} {@.id1})	**ANY**

c) If **AUTOMATIC TAGS** is in effect for the ASN.1:1994 module, apply 22.5-22.7 of ITU-T Rec. X.680 | ISO/IEC 8824-1, which describes how automatic tagging is applied to the components of a **SET**, **SEQUENCE** or **CHOICE** type, and remove the syntax "**AUTOMATIC TAGS**" from the module definition statement.

d) If an open type is constrained using the TypeConstraint subtype notation, remove the constraint, since **ANY** and **ANY DEFINED BY** cannot be constrained with anything but an **ANY** or **ANY DEFINED BY** in ASN.1:1990.

e) If the definition of an **ENUMERATED** type uses the syntax "**identifier**"

for its EnumerationItem, change it to "**identifier(number)**". For example, change:

```
ENUMERATED {a, b, c, d}
```

to

```
ENUMERATED {a(0), b(1), c(2), d(3)}
```

f) Remove all occurrences of extension markers (i.e. "**...**"). For example, change:

```
SEQUENCE {
        i       IA5String,
        b       BOOLEAN,
        ...
        }
```

to

```
SEQUENCE {
        i       IA5String,
        b       BOOLEAN
        }
```

g) Remove any occurrence of **EXTENSIBILITY IMPLIED** from the module definition statement.

h) Remove all whitespace and newline characters from within hstrings and bstrings, and remove all newline characters from within cstrings. For example, change:

```
b BIT STRING      ::= '0001 1100 1101 0110 1110 0111 1101 0101'B
o OCTET STRING    ::= '8F3CE483 0192B345 932D5EF2 8AA3E700'H
p PrintableString ::= "Hello,
                       world!"
```

to

```
b BIT STRING      ::= '00011100110101101110011111010101'B
o OCTET STRING    ::= '8F3CE4830192B345932D5EF28AA3E700'H
p PrintableString ::= "Hello,world!"
```

i) Convert all CharacterStringList notations to their cstring equivalent. For example, change:

```
name PrintableString ::= {"This is a long string, that is
                           spread across two lines"}
```

to

```
name PrintableString ::= "This is a long string, that is spread across two lines"
```

j) Convert any value set references into references to constrained types. For example, change:

```
Ages INTEGER ::= {1 | 4 | 7..20}
```

to

```
Ages ::= INTEGER (1 | 4 | 7..20)
```

NOTE 2 – It is preferable to use constrained types instead of value sets unless parameterization and information object classes are being heavily used, for this is typically when value sets are particularly useful.

k) Convert all occurrences of the **INSTANCE OF** type to their equivalent **SEQUENCE** type. For example, change:

```
A ::= INSTANCE OF REGISTERED-AS
```

to

```
A ::= SEQUENCE {
        type-id                 OBJECT IDENTIFIER,
        value                   [0]    ANY DEFINED BY type-id
}
```

l) Remove the identifiers "**mantissa**", "**base**" and "**exponent**" from any value of type **REAL**, and replace any inner type constraints present on **REAL** types with a comment. For example, change:

```
ten REAL ::= {mantissa 1, base 10, exponent 1}

DecimalReal ::= REAL (WITH COMPONENTS {..., base 10})
```

to

```
ten REAL ::= {1, 10, 1}

DecimalReal ::= REAL    -- Shall be encoded as base 10
```

m) Change all occurrences of the value notation for **EXTERNAL** to its ASN.1:1990 equivalent. For example, change:

```
extern1990 EXTERNAL ::= {
        direct-reference        { 1 2 3 4 5 6 },
        indirect-reference      3,
        encoding                single-ASN1-type : IA5String : "hello"
}
```

to

```
extern1994 EXTERNAL ::= {
        identification  context-negotiation :  {
                presentation-context-id         3,
                transfer-syntax                 { 1 2 3 4 5 6 }
        },
        data-value              notation : IA5String : "hello"
}
```

n) Replace all ASN.1:1994 permitted alphabets with their ASN.1:1990 equivalents. For example, change:

```
UpperCaseAndSpaceOnly ::= PrintableString (FROM("A".."Z" | " "))
```

to

```
UpperCaseAndSpaceOnly ::= PrintableString (FROM("A" | "B" | "C" |
                        "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" |
                        "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" |
                        "T" | "U" | "V" | "W" | "X" | "Y" | "Z" ))
```

o) Change all ASN.1:1994 set expressions used in the subtype notation to their ASN.1:1990 equivalent. For example, change:

```
PartNumber ::= NumericString (SIZE(8) ^ FROM("0".."9"))
```

to

```
PartNumber ::= NumericString (SIZE(8)) (FROM("0"|"1"|"2"|"3"|"4"|
"5"|"6"|"7"|"8"|"9"))
```

p) Where o) cannot be done because it results in an infinite set, replace the portion of the ASN.1:1994 subtype notation that results in an infinite set with a comment. For example, change:

```
AllButZeroToTen ::= INTEGER(ALL EXCEPT (0..10))
```

to

```
AllButZeroToTen ::= INTEGER    -- all integer values except 0 - 10
```

Applying the above conversion instructions to the ASN.1:1994 **ExampleModule** in guideline 5 produces:

```
-- <ASN1.Version 1990 ExampleModule > --

ExampleModule {-- a valid object identifier value goes here --}

DEFINITIONS ::= BEGIN

Foo::= SEQUENCE {
      id1 OBJECT IDENTIFIER,
      syntax1 ANY DEFINED BY id1}

Bar::= SEQUENCE {
      id2 OBJECT IDENTIFIER,
      syntax2 ANY }

-- Note that in the ASN.1:1994 ExampleModule, syntax2 in Bar is defined as a
-- SEQUENCE OF, not as an open type, and so cannot be
-- converted to an ANY DEFINED BY.

firstExtensionId OBJECT IDENTIFIER ::= {1 3 17 103 10 1}

FirstExtensionInfo ::= PrintableString

-- Note firstExtensionId and FirstExtensionInfo are to be used with the
-- Foo type where firstExtensionId is the value of the id1
-- (OBJECT IDENTIFIER) which indicates that the syntax1 has the
-- syntax type of FirstExtensionInfo (PrintableString).

END
```

7) New clause 10

Add a new clause 10, following clause 9:

10 Conventions for ASN.1 and GDMO directives

This clause introduces conventions to clearly identify specification and user options associated with GDMO templates and related ASN.1 modules. This is accomplished through the use of in-line directives. These conventions could also be useful as ASN.1 and GDMO compiler directives. If these conventions are used, they do not require a specifier to modify the ASN.1 and GDMO specifications in order to use these directives. That is, the directives may be in the same body of texts as the ASN.1 modules or GDMO templates, or in some other associated specification. Also, if these conventions are used, they do not alter the ASN.1 and GDMO specification. That is, these directives do not change the syntax of the ASN.1 and GDMO specification. These conventions are not required but are recommended.

These directives are structured to meet several requirements:

- Input files with directives (i.e. ASN.1 modules, GDMO libraries, and/or distinct directive files) must be acceptable to ASN.1 compilers and GDMO tools/compilers that do not recognize directives.
- Input files without directives must be acceptable to ASN.1 compilers and GDMO tools/compilers.

NOTE – These conventions permit extensions through the use of implementation specific directives.

Each directive specification is a structured comment that contains a single directive, which consists of a keyword qualified by a scope, and followed by zero or more operands. Case is significant. Thus, a directive is treated as a comment by any tool or compiler that does not support these directives.

The following conventions are used to describe directives:

- Text shown in bold (e.g. **`--<`** or **`ASN1`**) must be entered exactly as shown, without any added whitespace or change in case from that shown.
- Text shown in italics (e.g. *`keyword`*) should be replaced with appropriate text.
- Optional directive elements are enclosed in square brackets (e.g. **`[operands]`**).
- A directive element that may be repeated (any number of times) is followed by three periods (`...`) (e.g. **`[operands]`** `...`).

The general format of a directive is:

```
--< directive >--
```

where directive is:

```
Scope.keyword [operand ] [ , operand ] ...
```

That is, a directive is introduced by two consecutive hyphens and a less than symbol (**--<**), and terminated by a greater than symbol and two consecutive hyphens (**>--**). There must not be any whitespace between the hyphens and the less than or greater than symbols. Thus, a directive is treated as an ASN.1 or GDMO comment by any compiler that does not support these directives. A directive cannot contain any ASN.1 comments.

Each `--<  >--` construction is a structured comment that contains a single directive, which consists of a keyword qualified by a scope, and followed by zero or more operands. Case is significant.

Directives may be continued onto additional lines, with the first non-whitespace characters being two consecutive hyphens (`--`) on the continued line(s).

Operands are delimited by whitespace (one or more consecutive space or tab characters) for dissimilar items, and a comma for a list of similar items (such as working set names). White space (zero or more space or tab characters) may occur before or after other elements in a directive. In this context, carriage return, new line, or vertical tab characters are NOT considered whitespace, and are not allowed.

10.1 Conventions for ASN.1 directives

For an ASN.1 directive, the following additional conventions apply.

A directive may be in the same file as the ASN.1 item. When placed in the same file, the directive may be outside the scope of an ASN.1 module (before the start of a module or after the end of a module). A directive may occur within the body of a module, wherever whitespace is allowed. When placed within a module, each directive must occur before the ASN.1 item to which it applies.

For an ASN.1 directive, the Scope symbol is **`ASN1`**. Additionally, other scope symbols may be defined (e.g. implementation assigned).

The keyword can be one of the following:

- **`Version.`**

In general, an operand may be a:

- text string (without white space, comma, **`--`** or **`>`**);
- number string (without white space, comma, **`--`** or **`>`**);
- cstring ("xxxxx") all on same line;
- { *ObjectIdentifier* };
- potentially other ASN.1 constructions, e.g. bstring or hstring.

10.1.1 Version directive

The **`Version`** directive is used to indicate if an ASN.1 module is written according to the ASN1:1990 or ASN1:1994 standard.

This directive has the following format:

`--<ASN1.Version` *`version moduleName`* **`[`** *`oid`* **`]>--`**

The elements shown in bold (e.g. **`ASN1.Version`**) are to be written as shown, and the elements in italics (e.g. *`version`*) are to be replaced as follows:

- *`version`* – Either **1990** or **1994** or **1990, 1994**.
- *`moduleName`* – The name of an ASN.1 module.
- `oid` – An optional ASN.1 Object Identifier value, used to unambiguously identify an ASN.1 module.

Only one **`Version`** directive should be provided for an ASN.1 module. The **`Version`** directive **1990, 1994** means that the ASN.1 module asserts conformance to both ASN.1:1990 and ASN.1:1994; a compiler may use its knowledge of either of these versions, or the common subset of both. If a **`Version`** directive is not provided for an ASN.1 module, an implementation may use some other way to identify the module version, such as command line compiler option, or the recognition of syntax that is unique to a particular version (such as an ASN1:1990 macro, or an ASN1:1994 information object).

Examples:

```
--<ASN1.Version 1990 Attribute-ASN1Module
--      {joint-iso-itu-t ms(9) smi(3) part2(2) asn1Module(2) 1} >-

--<ASN1.Version 1994 SMModule
--      {joint-iso-itu-t ms(9) smi(3) part4(4) asn1Module(2) 2} >--
```

10.2 Conventions for GDMO directives

A GDMO directive has the following additional conventions.

A directive cannot contain any ASN.1 comments.

A directive may be in a file different than the file that contains the GDMO template the directive refers to, or it may be in the same file as the GDMO template. When placed in the same file, the directive may be outside the scope of a GDMO document (before the start of a document or after the end of a document). A directive may occur within the body of a document, wherever whitespace is allowed.

When placed within a document, each directive must occur before the GDMO template to which it applies. For a particular GDMO template, there may be at most one particular directive for that template, but there may be different directives for the same template. For example, the same template may be referenced by both Nickname and Working Set directives, but only one each. There may be Nickname directives for other ASN.1 items.

The Scope symbol is **`GDMO`**. Additionally, other scope symbols may be defined (e.g. implementation assigned).

The keyword can be one of those defined here:

- **`Alias;`**
- **`Document;`**
- **`EndDocument;`**
- **`Version.`**

In general, an operand may be a:

- text string (without white space, comma, **`--`** or **`>`**);
- number string (without white space, comma, **`--`** or **`>`**);
- cstring ("xxxxx") all on same line;
- { *ObjectIdentifier* };
- potentially other ASN.1 constructions, e.g. bstring or hstring.

10.2.1 Alias directive

The **`Alias`** directive is used to provide alternative or alias identifiers for a GDMO document. These aliases are used to reconcile references between GDMO documents, where short or inconsistent document identifiers are used.

The format of the directive is:

`--<GDMO.Alias` *`documentIdentifier documentAlias`*

`-- [ ,` *`documentAlias`* **`] ... >--`**

The elements shown in bold (e.g. **`GDMO.Alias`**) are to be written as shown, and the elements in italics (e.g. *`documentIdentifier`*) are to be replaced as described below.

Any number of documentAlias items may be provided, each separated with a comma.

- *`documentIdentifier`* – The identifier of a GDMO document, either in the form of a string, enclosed in quotes ("), or an Object Identifier, enclosed in curly braces ({ and }).
- *`documentAlias`* – An alternative identifier of a GDMO document, either in the form of a string, enclosed in quotes ("), or an Object Identifier, enclosed in curly braces ({ and }).

Examples:

```
--<GDMO.Alias   "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:1992"
--  "Rec. X.721 | ISO/IEC 10165-2:1992",
--  "CCITT Rec. X.721 | ISO/IEC 10165-2 ",
--  "Rec. X.721| ISO/IEC 10165-2",
--  "CCITT Rec. X.721 (1992)",
--  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:1992",
--  "DMI">--
--<GDMO.Alias  "Recommendation M.3100:1992"
--  "Rec. M.3100:1992",
--  "M.3100:1992",
--  "M.3100">--
```

10.2.2 Document directive

The **Document** directive provides the identifier of a GDMO "document", where the identifier is either a character string, or an Object Identifier, or both. The format of the directive can take one of three forms:

--<GDMO.Document *documentString* **>--**

--<GDMO.Document *documentOID* **>--**

--<GDMO.Document *documentString documentOID* **>--**

The elements shown in bold (e.g. **GDMO.Document**) are to be written as shown, and the elements in italics (e.g. *documentString*) are to be replaced as follows:

- *documentString* – The character string that identifies a GDMO document, enclosed in quotes (").
- *documentOID* – The ASN.1 Object Identifier assigned to a GDMO document.

A **Document** directive must come before the first GDMO template or ASN.1 module making up the document. That is, a GDMO document is considered to be composed of all the GDMO templates and ASN.1 modules following a **Document** directive, up to the corresponding **EndDocument** directive, to the next **Document** directive, or to the end of the file that contains the GDMO text.

The following rules for storing GDMO text in files have the corresponding directives:

- Each GDMO document should have a **Document** directive to provide the "correct" name of the document. If a **Document** directive is not provided, the name of the document is not defined, or is provided through some other mechanism.
- A **Document** directive should be in the same file as the GDMO text to which it applies.

The same GDMO document name may appear in several **Document** directives, to apply to several units of GDMO text, such as distinct files. In this case, all of the GDMO text in the different files are considered part of the same GDMO document. This is similar to the C++ namespace, which allows several distinct header files to contain material defined in the same namespace.

In general, there should be a single GDMO document per input file. The only case a file may contain more that one GDMO document is when the file has appropriate **Document** and **EndDocument** directives.

Whitespace (one or more consecutive space or tab characters) is not considered significant within a *documentIdentifier* or *documentAlias*.

Examples:

```
--<GDMO.Document  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2:1992">--
--<GDMO.Document  "Recommendation M.3100:1992">--
--<GDMO.Document  "OP1 Library Vol. 4">--
--<GDMO.Document  {iso(1) 2 124 360501 15 13 1} >--
--<GDMO.Document  "IIMC MIB Translation" {iso(1) 2 124 360501 15 13 1} >--
```