



INTERNATIONAL STANDARD ISO 8879:1986
TECHNICAL CORRIGENDUM 2

Published 1999-11-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information processing — Text and office systems — Standard Generalized Markup Language (SGML)

TECHNICAL CORRIGENDUM 2

Traitement de l'information — Systèmes bureautiques — Langage normalisé de balisage généralisé (SGML)

RECTIFICATIF TECHNIQUE 2

ITeH STANDARD PREVIEW
(standards.iteh.ai)

Technical Corrigendum 2 to International Standard ISO 8879:1986 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

<https://standards.iteh.ai/catalog/standards/sist/9595ffc6-6f71-4b5f-b5db-912cc6b4e3f4/iso-8879-1986-cor-2-1999>

All occurrences of "element declaration" in this International Standard are changed to "element type declaration".

All occurrences of "element set" in this International Standard are changed to "element type set".

Replace 4.223 with:

4.223 owner identifier: The portion of a public identifier that identifies its owner.

Note: The owner of a public identifier is not necessarily the owner of the object that it identifies.

Add the following normative annex K and informative annex L to ISO 8879.

Annex K (normative) Web SGML Adaptations

This annex remedies defects revealed by the multiple adaptations of SGML for the World Wide Web (WWW), intranets, and extranets. The annex corrects errors, resolves ambiguities for which there is a clear resolution that does not cause existing conforming documents to become non-conforming, and provides a choice of alternative resolutions for other ambiguities. Although motivated by the World Wide Web, applicability of this annex extends to all uses of SGML.

This annex makes reference to groves and property sets, which are defined in the SGML Extended Facilities of the 2nd Edition of HyTime (ISO/IEC 10744) and also in DSSSL (ISO/IEC 10179).

K.1 Conformance

The decision to conform to this annex is made for individual documents and is indicated in the SGML declaration, as described below (K.3.2). An SGML system need not support this annex in order to be a conforming SGML system.

This annex is organized as a set of replacement and new syntax productions and text, and is phrased in terms of modifications to be made to the body of this International Standard, though the numbers of the affected clauses are not necessarily cited. However, these modifications are applicable only when conforming to this annex.

An SGML parser that supports this annex shall be able to parse conforming SGML documents that do not claim conformance to it. The parsing of such documents must produce the same grove as would a conforming SGML parser that does not support this annex. Error reporting for such documents, however, is with respect to the standard as modified by this annex; errors with respect to the unmodified standard need not be reported.

K.2 Definitions

K.2.1 Definitions related to document type declarations

K.2.1.1 DTD declarations:

Markup declarations that occur in a document type declaration subset.

K.2.1.2 external subset:

The portion of a document type declaration subset located within the external subset entity.

K.2.1.3 internal subset:

The portion of a document type declaration subset that occurs between the dso and dsc of a document type declaration.

If the dso and dsc are omitted, the internal subset is empty (rather than non-existent).

K.2.1.4 external subset entity:

The entity that is declared by the external identifier parameter of a document type declaration and referenced (implicitly) at the end of the internal subset.

K.2.2 Definitions related to document instances

K.2.2.1 fully-declared document instance:

A document instance whose associated document type declaration contains sufficient markup declarations to express all of the DTD properties required for the instance by the body of this International Standard, and that does not contain DTD data entities that modify those properties.

The document type declaration itself could be an implied declaration, as provided in clause K.4.9.

Note 1: An SGML declaration requires document instances to be fully-declared if it specifies IMPLYDEF ATTLIST NO and ELEMENT NO ENTITY NO NOTATION NO. A system should offer means, such as a parameter to the invocation of processing, to request validation of whether an instance is fully-declared even when the SGML declaration does not require it to be.

A document instance that is not fully-declared must be either a fully-tagged document instance or an amply-tagged document instance.

K.2.2.2 fully-tagged document instance:

A document instance in which a start-tag with a generic identifier, and an end-tag, are present for every element, and the attribute name is present in every attribute specification in the start-tag.

Note 2: An SGML declaration requires document instances to be fully-tagged if it specifies DATATAG NO, RANK NO, OMITTAG NO, SHORTTAG STARTTAG EMPTY NO, and SHORTTAG ATTRIB OMITNAME NO. A system should offer means, such as a parameter to the invocation of processing, to request validation of whether an instance is fully-tagged even when the SGML declaration does not require it to be.

K.2.2.3 type-valid document instance:

A document instance that conforms to such DTD properties as are required for it by the body of this International Standard and that are expressed in its associated document type declaration, either by markup declarations, DTD data entities, or both.

K.2.2.4 amply-tagged document instance:

A document instance whose use of markup minimization does not require access to a document type declaration.

Note 3: An SGML declaration requires document instances to be amply-tagged, insofar as they use minimization, if it specifies DATATAG NO, RANK NO, SHORTTAG ATTRIB OMITNAME NO, and either IMPLYDEF ELEMENT NO or IMPLYDEF ELEMENT ANYOTHER. A system should offer means, such as a parameter to the invocation of processing, to request validation of whether an instance is amply-tagged even when the SGML declaration does not require it to be.

Note 4:

End-tags can be omitted in an amply-tagged document instance, even for element types that have been declared by default as provided for in K.3.7.

More precisely, an amply-tagged document instance can omit the end-tag for an element of an implicitly-declared type by satisfying any of the three conditions -- a), b), or c) -- specified in 7.3.1.2. However, condition c) can be satisfied only when "IMPLYDEF ELEMENT ANYOTHER" is specified in the SGML declaration and the element is followed by another element of the same type.

The following amply-tagged document instance illustrates these possibilities when "IMPLYDEF ELEMENT ANYOTHER" has been specified and there are no DTD declarations:

```
<chapter>
<title>Chapter title</title>
<list>
<item>First item in the list
<item>Second item in the list
</list>
```

The end-tag for the first item in the list can be omitted because of condition c): This element is followed by an element that is not allowed in its content--in particular, another item.

The end-tag for the second item in the list can be omitted because of condition b): This element is followed by the end-tag of another open element (the list).

The end-tag for the chapter can be omitted because of condition a): This element is followed by the end of the SGML document entity.

K.2.2.5 immediately recursive element:

An element whose immediate content includes a child subelement of the same type.

Note 5: In the following fully-tagged text, the first section element is immediately recursive while the first list element is not.

```
<section>
<title>The title</title>
<section>
<p>The data.</p>
</section>
</section>

<list>
<item>First outer list item</item>
<item>Second outer list item, which contains an inner list.
<list>
<item>Inner list item</item>
</list>
</item>
</list>
```

Note 6: A system should offer means, such as a parameter to the invocation of processing, to request validation that an instance contains no immediately recursive elements even when the DTD would permit them.

K.2.3 Definitions related to entity constraints

Note 7: An SGML system that supports unconstrained SGML documents must be able to parse DTD declarations and resolve both internal and external entity references. If it continues parsing (as a form of error-recovery) after failing to access a referenced entity, the results are unpredictable. Observing one or more of the entity constraints defined in this International Standard may cause a document to be more amenable to processing by a simpler SGML system, or in an environment (such as a network) where access to external entities may be slow or unreliable.

K.2.3.1 integrally-stored document instance:

A document instance in which every element and marked section ends in the entity in which it begins.

Note 8: This constraint makes it possible, as a form of error-recovery, for parsing to continue in a fully-tagged document instance after a failure to access a referenced entity. The resulting grove will be the same for the parsed text, except for the tree addresses of younger siblings of the nodes in the inaccessible entity. The constraint also has implementation benefits for editors, "lazy" replacement of entity references, and sharing of grove portions when entities are reused.

K.2.3.2 reference-free document:

An SGML document that has no entity references other than references to predefined data character entities.

Note 9: A reference-free document can be parsed by conforming SGML systems that cannot resolve entity references.

K.2.3.3 external-reference-free document:

An SGML document that has no external entity references.

Note 10: An external-reference-free document could have attribute values that contain names of external entities or that otherwise might cause an application to access an external entity by means other than entity references.

Note 11: External-reference-free documents can be parsed by systems that cannot resolve external entity-references.

K.2.4 Other definitions

K.2.4.1 predefined data character entity:

A general entity, associated with a character number in the syntax-reference character set, that is used to reference significant SGML characters as data.

Note 12: In order to allow delimiter escaping when parsing without respect to DTD declarations, there should be a predefined data character entity for the first character of each delimiter string that can be recognized in a mode where data can occur.

K.2.4.2 white space:

The characters assigned to the SEPCHAR, SPACE, RE, and RS functions.

K.2.4.3 mandatorily empty element:

An element of a type declared EMPTY, or that is forced to be EMPTY by an explicit content reference attribute.

K.2.5 Definitions related to DTD notations

K.2.5.1 DTD notation:

A data notation that is capable of expressing DTD properties.

Note 13: This International Standard provides markup declarations as the means of expressing DTD properties. It also allows the use of DTD notations, but does not define any.

Note 14: A DTD notation can also be capable of expressing other information. For example, it could express constraints on a document's data content or structure that cannot be expressed by markup declarations but could be validated by an application. However, it is not a reportable markup error if the document fails to conform to such additional constraints.

Note 15: DTD notations are used in DTD data entities.

<https://standards.iteh.ai/catalog/standards/sist/9595ffc6-6f71-4b5f-b5db-912cc6b4e3f4/iso-8879-1986-cor-2-1999>

K.2.5.2 DTD properties:

Classes and properties that are specifiable by markup declarations. They are defined by the following grove plan:

```
<!DOCTYPE grovplan PUBLIC "ISO/IEC 10744:1997//DTD Grove
Plan//EN">
<grovplan propset=SGMLProp id=dtdprops>
<title>DTD Properties Grove Plan</title>
<desc>
Classes and properties that are specifiable by DTD declarations. They
are needed to parse and validate a document instance.
</desc>
<inclmod>
prlgabs0 prlgabs1 dtgabs rankabs srabs subdcabs
fpiabs arcabs fsiabs dafecabs gadcabs pelement
</inclmod>
<inclclas>
sgmldoc doctpdcl
</inclclas>
<omitprop classes="sgmldoc">
appinfo epilog
</omitprop>
</grovplan>
```

K.2.5.3 Parameter entity (4.225)

An entity that is either the external subset entity, or that is declared with a parameter entity name.

K.2.5.4 DTD data entity:

An external parameter entity whose declaration includes a notation name.

Note 16: The notation name should be that of a DTD notation and the entity content should describe DTD properties.

K.3 SGML declaration

[171] **SGML declaration** =
"<!SGML", ps+,
(SGML declaration reference | SGML declaration body),
ps*, ">"

In order for SGML documents to be self-identifying, it is strongly recommended that all conforming SGML documents contain one of the forms of SGML declaration.

K.3.1 SGML declaration reference

[171.1] **SGML declaration reference** =
name, external identifier?
where:
"name" is a name in the reference concrete syntax.
"external identifier" must reference an SGML declaration body. If
omitted, the external identifier is "SYSTEM".

An external identifier, if it is a formal public identifier, should use "SD" as the public text class to identify an SGML declaration body, rather than one of those listed in 10.2.2.1.

Note 17: For example:

```
<!SGML HTML3.2 PUBLIC "-//IDN W3C.ORG//SD HTML Version 3.2//EN">
```

In 10.2.2.1 replace the phrase "The name must be" with "The name should be".

K.3.2 SGML declaration body

[171.2] **SGML declaration body** =
minimum literal, ps+,
document character set, ps+,
capacity set, ps+,
concrete syntax scope, ps+,
concrete syntax, ps+,
feature use, ps+,
application-specific information,
(ps+, added requirements)?

The minimum data of the minimum literal shall be one of the following:

- "ISO 8879:1986"
- "ISO 8879:1986 (ENR)"
- "ISO 8879:1986 (WWW)"

The ENR suffix indicates that the document conforms to Annex J of this International Standard.

The WWW suffix indicates that the document conforms to Annex K (this annex) of this International Standard, which incorporates by reference the provisions of Annex J of this International Standard and supersedes any contradictory provisions elsewhere in this International Standard.

Certain parameters of the SGML declaration body are omissible in order to allow documents to declare conformance to this annex by a simple change to the minimum data. An omitted parameter has no effect on the document; that is, with respect to the subject of the omitted parameter, the document is the same as if it did not conform to this annex.

Note 18: Nevertheless, it is recommended that all parameters be specified.

K.3.3 Capacity Set

```
[180] capacity set =
"CAPACITY", ps+,
(("PUBLIC", ps+, public identifier)|
"NONE"|
("SGMLREF", (ps+, name, ps+, number)*))
```

NONE indicates that no capacities are specified.

Specifying NONE does not require a system to support capacities greater than those specified in its system declaration.

K.3.4 Concrete syntax

```
[182] concrete syntax =
"SYNTAX", ps+,
(public concrete syntax |
(shunned character number identification, ps+,
syntax-reference character set, ps+,
function character identification, ps+,
naming rules, ps+,
delimiter set, ps+,
reserved name use, ps+,
quantity set,
(ps+, predefined data character entities?))
```

ISO 8879:1986/Cor 2:1999

K.3.4.1 Quantity Set

```
[194] quantity set =
"QUANTITY", ps+,
("NONE"|
("SGMLREF", (ps+, name, ps+, number)*))
```

NONE indicates that no quantities are specified.

Specifying NONE does not require a system to support quantities greater than those specified in its system declaration.

K.3.4.2 Predefined data character entities

```
[194.1] predefined data character entities =
"ENTITIES", ps+,
("NONE"|(parameter literal, ps+, character number)+)
```

Each interpreted "parameter literal" must be a valid general entity name in the syntax being defined, and is associated with a character number in the syntax-reference character set. When the named entity is referenced, the replacement text is a numeric character reference to the corresponding character. Predefined data character entities are treated as though defined at the start of the internal subset of all documents in which the concrete syntax is used.

Note 19: For example:

```
ENTITIES "amp" 38 "lt" 60 "gt" 62 "quot" 34 "apos" 39
```

K.3.5 Markup minimization features

[196] **markup minimization features** =
 "MINIMIZE", ps+,
 "DATATAG", ps+, ("NO"|"YES"), ps+,
 "OMITTAG", ps+, ("NO"|"YES"), ps+,
 "RANK", ps+, ("NO"|"YES"), ps+,
 "SHORTTAG", ps+, ("NO"|"YES")|
 (start-tag options, ps+, end-tag options, ps+, attribute-options)),
 (ps+, empty element ending rules,
 ps+, implied default declarations)?

Note 20: Use of an enabled markup minimization feature may be affected by the operation of other provisions of this International Standard, including other markup minimization features.

K.3.5.1 SHORTTAG start-tag options

[196.1] **start-tag options** =
 "STARTTAG", ps+,
 "EMPTY", ps+, ("NO"|"YES"), ps+,
 "UNCLOSED", ps+, ("NO"|"YES"), ps+,
 "NETENABL", ps+, ("NO"|"ALL"|"IMMEDNET")
 where:
 EMPTY YES enables empty start-tags.
 UNCLOSED YES enables unclosed start-tags.
 NETENABL ALL enables NET-enabling start-tags for beginning all elements that are permitted to have end-tags. An element so begun may, but need not, be ended with a null end-tag.
 NETENABL IMMEDIATE ("immediate NET") enables NET-enabling start-tags for beginning elements that are permitted to have end-tags and that have no text between the start-tag and the end-tag, provided that each element so begun is ended with a null end-tag.

Note 21: An element with no text between its start-tag and end-tag is not necessarily a mandatorily empty element.

Note 22: A null end-tag can be used for a mandatorily empty element only when EMPTYNRM YES is specified (see K.3.6).

Note 23: For example, if the **nestc** is "/" and the **net** is ">", and "img" is either not a mandatorily empty element or is a mandatorily empty element and EMPTYNRM YES is specified, then:

can be used as an alternative to

K.3.5.2 SHORTTAG end-tag options

[196.2] **end-tag options** =
 "ENDTAG", ps+,
 "EMPTY", ps+, ("NO"|"YES"), ps+,
 "UNCLOSED", ps+, ("NO"|"YES")
 where:
 EMPTY YES enables empty end-tags.
 UNCLOSED YES enables unclosed end-tags.

K.3.5.3 SHORTTAG attribute options

[196.3] **attribute options** =
 "ATTRIB", ps+,
 "DEFAULT", ps+, ("NO"|"YES"), ps+,
 "OMITNAME", ps+, ("NO"|"YES"), ps+,
 "VALUE", ps+, ("NO"|"YES")

where:

DEFAULT YES enables attribute value defaulting (7.9.1.1).

OMITNAME YES enables attribute names and **vi** to be omitted for unique NMTOKEN values (7.9.1.2).

VALUE YES enables some attribute values to be specified without delimiters, rather than as literals (7.9.3.1).

Note 24: DEFAULT NO does not bar default values from attribute definition list declarations.

K.3.6 Empty element ending rules

[196.4] **empty element ending rules** =

"EMPTYNRM", ps+, ("NO"|"YES")

where:

EMPTYNRM YES applies normal rules for the presence of end-tags, including markup minimization rules, to elements of a type declared EMPTY, or that are forced to be EMPTY by an explicit content reference attribute (7.3.).

Note 25: Specifying EMPTYNRM YES applies to mandatorily empty elements the same rules about the presence of end-tags that apply to other kinds of elements.

K.3.7 Implied default declarations

[196.5] **implied default declarations** =

"IMPLYDEF", ps+,

"ATTLIST", ps+, ("NO"|"YES"), ps+,

"DOCTYPE", ps+, ("NO"|"YES"), ps+,

"ELEMENT", ps+, ("NO"|"YES"|"ANYOTHER"), ps+,

"ENTITY", ps+, ("NO"|"YES"), ps+,

"NOTATION", ps+, ("NO"|"YES")

where:

DOCTYPE YES means that an implied document type declaration includes an implied declaration for an external subset entity (see K.4.9); and YES for the other parameters allows information of the specified type to be used in a fully-tagged or amply-tagged document instance without an explicit declaration. Instead, a declaration is implied, as follows:

ATTLIST YES means an undeclared attribute is declared as: CDATA #IMPLIED

ELEMENT YES means an undeclared element type is declared as: - 0 ANY

ELEMENT ANYOTHER means an undeclared element type is declared as: - 0 ANY but immediately recursive elements of that type are prohibited.

ENTITY YES means an undeclared general entity is declared as: SYSTEM

NOTATION YES means an undeclared notation is declared as: SYSTEM

Note 26: IMPLYDEF DOCTYPE YES implies only a declaration for an external subset entity. Other declarations may be implied if needed, and if permitted by the applicable IMPLYDEF parameters.

When IMPLYDEF ELEMENT ANYOTHER and OMITTAG YES are specified, the end-tag of an element of an undeclared type can be omitted from an amply-tagged instance when the element is followed by the start of an element of the same type.

When IMPLYDEF ENTITY YES is specified, #DEFAULT cannot be specified as a general entity name.

Implicitly declared definitions occur in the grove in the order in which it is necessary to imply their declarations. No attribute assignment node is created for an attribute unless the attribute is specified.

Note 27: An implied declaration is not necessarily the same as an explicit declaration for the same object that was ignored during parsing (Note 38). Nor does it constrain any explicit declaration that might be created for that object as a result of processing (for example, when generating an explicit DTD for a document that has none).

K.3.8 Other features

[198] **other features** =
 "OTHER", ps+,
 "CONCUR", ps+, ("NO"|"YES", ps+, number)), ps+,
 "SUBDOC", ps+, ("NO"|"YES", ps+, number)), ps+,
 "FORMAL", ps+, ("NO"|"YES"),
 (urn feature, keepsre feature, validity feature, entities feature)?

K.3.8.1 Universal Resource Names

[198.1] **urn feature** =
 ps+, "URN", ps+, ("NO"|"YES")
 where:
 URN YES means public identifiers are interpreted according to the
 applicable Internet Engineering Task Force RFC2141 governing
 Universal Resource Names.

If both URN and FORMAL are YES, public identifiers are interpreted either as formal public identifiers or as URNs.

K.3.8.2 White space in content

[198.2] **keepsre feature** =
 ps+, "KEEPSRE", ps+, ("YES"|"NO")
 where:
 KEEPSRE YES means clause 7.6.1 does not apply.

If KEEPSRE YES is specified, all white space in mixed content is included in the grove as datachar nodes and all white space in element content is included in the grove as s separator nodes.

Note 28: This option does not affect delimited strings, such as attribute value literals, which have their own rules for normalizing white space (and which, in any case, do not occur in content).

K.3.8.3 Assertions

[198.3] **validity feature** = ps+,
 "VALIDITY", ps+, ("NOASSERT"|"TYPE")
 where:
 NOASSERT makes no validity assertion.
 TYPE asserts that document instances are type-valid.
 If the parameter is omitted, TYPE is assumed.

[198.4] **entities feature** = ps+,
 "ENTITIES", ps+, ("NOASSERT"|
 ("REF", ps+, ("NONE"|"INTERNAL"|"ANY"), ps+,
 "INTEGRAL", ps+, ("NO"|"YES")))

where:
 NOASSERT makes no validity assertion.
 REF asserts the document either has unconstrained entity references (ANY),
 is external-reference-free (INTERNAL), or is reference-free (NONE).
 INTEGRAL YES asserts document instances are integrally-stored.
 If the parameter is omitted, NOASSERT is assumed.

It is a reportable markup error if a document is less constrained than is asserted.

Note 29: For example, if an otherwise conforming document incorrectly asserts that it is integrally-stored, the document is non-conforming. Had the assertion not been made, the document would have been conforming.

Note 30: A system should offer means, such as a parameter to the invocation of processing, to check whether a particular VALIDITY or ENTITIES assertion could correctly be made for a document, even when the SGML declaration does not make that assertion.

Note 31: To satisfy the classical requirement for SGML conformance, a document instance must be fully-declared as well as type-valid.