
International Standard



8879

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

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

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

First edition — 1986-10-15

ITeH STANDARD PREVIEW
(standards.iteh.ai)

ISO 8879:1986

<https://standards.iteh.ai/catalog/standards/sist/bbf25d89-cdc0-4eac-8ec4-00cfb29943ad/iso-8879-1986>

UDC 681.3.06

Ref. No. ISO 8879-1986 (E)

Descriptors : data processing, documentation, logical structure, programming (computers), artificial languages, programming languages.

Price based on 155 pages

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 8879 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

Contents

0 Introduction	1
0.1 Background	1
0.2 Objectives	2
0.3 Organization	3
1 Scope	4
2 Field of Application	4
3 References	5
4 Definitions	5
5 Notation	20
5.1 Syntactic Tokens	21
5.2 Ordering and Selection Symbols	21
6 Entity Structure	21
6.1 SGML Document	21
6.2 SGML Entities	21
6.2.1 S Separator	22
6.2.2 Entity End	22
6.2.3 Implied SGML Declaration	22
6.3 Non-SGML Data Entity	22
7 Element Structure	22
7.1 Prolog	22
7.2 Document Element	22
7.2.1 Limits	22
7.3 Element	22
7.3.1 Omitted Tag Minimization	22
7.3.1.1 Start-tag Omission	22
7.3.1.2 End-tag Omission	23
7.3.2 Data Tag Minimization	23
7.3.3 Quantities	23
7.4 Start-tag	23
7.4.1 Minimization	23
7.4.1.1 Empty Start-tag	23
7.4.1.2 Unclosed Start-tag	24
7.4.1.3 NET-enabling Start-tag	24
7.4.2 Quantities	24
7.5 End-tag	24
7.5.1 Minimization	24
7.5.1.1 Empty End-tag	24
7.5.1.2 Unclosed End-tag	24
7.5.1.3 Null End-tag	24
7.6 Content	24
7.6.1 Record Boundaries	25
7.7 Document Type Specification	25
7.7.1 General Entity References	25
7.7.2 Parameter Entity References	25
7.8 Generic Identifier (GI) Specification	26
7.8.1 Rank Feature	26
7.8.1.1 Full Generic Identifier	26
7.8.1.2 Rank Stem	26
7.9 Attribute Specification List	26

7.9.1	Minimization	26
7.9.1.1	Omitted Attribute Specification	26
7.9.1.2	Omitted Attribute Name	26
7.9.2	Quantities	26
7.9.3	Attribute Value Specification	26
7.9.3.1	Minimization	27
7.9.4	Attribute Value	27
7.9.4.1	Syntactic Requirements	27
7.9.4.2	Fixed Attribute	27
7.9.4.3	General Entity Name	27
7.9.4.4	Notation	27
7.9.4.5	Quantities	27
8	Processing Instruction	27
8.1	Quantities	28
9	Common Constructs	28
9.1	Replaceable Character Data	28
9.2	Character Data	28
9.2.1	SGML Character	28
9.2.2	Function Character	28
9.3	Name	28
9.3.1	Quantities	28
9.4	Entity References	28
9.4.1	Quantities	28
9.4.2	Limits	28
9.4.3	Obfuscatory Entity References	29
9.4.4	Named Entity Reference	29
9.4.5	Reference End	29
9.4.6	Short Reference	29
9.4.6.1	Equivalent Reference String	29
9.5	Character Reference	30
9.6	Delimiter Recognition	30
9.6.1	Recognition Modes	30
9.6.2	Contextual Constraints	31
9.6.3	Order of Recognition	32
9.6.4	Delimiters Starting with the Same Character	32
9.6.5	Short References with Blank Sequences	32
9.6.5.1	Quantities	32
9.6.6	Name Characters	32
9.7	Markup Suppression	32
9.8	Capacity	32
10	Markup Declarations: General	34
10.1	Parts of Declarations	34
10.1.1	Parameter Separator	34
10.1.2	Parameter Literal	34
10.1.2.1	Quantities	34
10.1.3	Group	34
10.1.3.1	Quantities	34
10.1.4	Declaration Separator	34
10.1.5	Associated Element Type	35
10.1.6	External Identifier	35
10.1.6.1	Quantities	35
10.1.6.2	Capacities	35
10.1.7	Minimum Literal	35
10.1.7.1	Quantities	35
10.2	Formal Public Identifier	35
10.2.1	Owner Identifier	35
10.2.1.1	ISO Owner Identifier	35
10.2.1.2	Registered Owner Identifier	35

10.2.1.3	Unregistered Owner Identifier	36
10.2.2	Text Identifier	36
10.2.2.1	Public Text Class	36
10.2.2.2	Public Text Description	36
10.2.2.3	Public Text Language	36
10.2.2.4	Public Text Designating Sequence	36
10.2.2.5	Public Text Display Version	37
10.3	Comment Declaration	37
10.4	Marked Section Declaration	37
10.4.1	Quantities	37
10.4.2	Status Keyword Specification	37
10.5	Entity Declaration	38
10.5.1	Entity Name	38
10.5.1.1	Quantities	38
10.5.1.2	Capacities	38
10.5.2	Entity Text	38
10.5.3	Data Text	38
10.5.4	Bracketed Text	39
10.5.4.1	Quantities	39
10.5.5	External Entity Specification	39
11	Markup Declarations: Document Type Definition	39
11.1	Document Type Declaration	39
11.2	Element Declaration	40
11.2.1	Element Type	40
11.2.1.1	Ranked Element	40
11.2.1.2	Quantities	40
11.2.2	Omitted Tag Minimization	40
11.2.3	Declared Content	40
11.2.4	Content Model	40
11.2.4.1	Connector	41
11.2.4.2	Occurrence Indicator	41
11.2.4.3	Ambiguous Content Model	41
11.2.4.4	Data Tag Group	41
11.2.4.5	Quantities	42
11.2.5	Exceptions	42
11.2.5.1	Inclusions	42
11.2.5.2	Exclusions	42
11.3	Attribute Definition List Declaration	42
11.3.1	Quantities	42
11.3.2	Attribute Name	42
11.3.3	Declared Value	43
11.3.4	Default Value	43
11.3.4.1	Quantities	43
11.3.4.2	Capacities	43
11.4	Notation Declaration	43
11.5	Short Reference Mapping Declaration	44
11.6	Short Reference Use Declaration	44
11.6.1	Use in Document Type Declaration	44
11.6.2	Use in Document Instance	44
11.6.3	Current Map	44
12	Markup Declarations: Link Process Definition	44
12.1	Link Type Declaration	44
12.1.1	Simple Link Specification	45
12.1.2	Implicit Link Specification	45
12.1.3	Explicit Link Specification	45
12.1.3.1	Limits	45
12.1.4	Link Type Declaration Subset	45
12.1.4.1	Parameter Entities	45
12.1.4.2	Link Attributes	45

12.1.4.3	Simple Link	45
12.2	Link Set Declaration	45
12.2.1	Source Element Specification	46
12.2.2	Result Element Specification	46
12.3	Link Set Use Declaration	46
12.3.1	Use in Link Type Declaration	46
12.3.2	Use in Document Instance	46
12.3.3	Current Link Set	46
13	SGML Declaration	47
13.1	Document Character Set	47
13.1.1	Character Set Description	47
13.1.1.1	Base Character Set	47
13.1.1.2	Described Character Set Portion	47
13.1.2	Non-SGML Character Identification	48
13.2	Capacity Set	48
13.3	Concrete Syntax Scope	48
13.4	Concrete Syntax	49
13.4.1	Public Concrete Syntax	49
13.4.2	Shunned Character Number Identification	49
13.4.3	Syntax-reference Character Set	50
13.4.4	Function Character Identification	50
13.4.5	Naming Rules	50
13.4.6	Delimiter Set	51
13.4.6.1	General Delimiters	51
13.4.6.2	Short Reference Delimiters	51
13.4.7	Reserved Name Use	51
13.4.8	Quantity Set	51
13.5	Feature Use	52
13.5.1	Markup Minimization Features	52
13.5.2	Link Type Features	52
13.5.3	Other Features	52
13.6	Application specific Information	53
14	Reference and Core Concrete Syntaxes	53
15	Conformance	53
15.1	Conforming SGML Document	53
15.1.1	Basic SGML Document	53
15.1.2	Minimal SGML Document	53
15.1.3	Variant Conforming SGML Document	53
15.2	Conforming SGML Application	53
15.2.1	Application Conventions	53
15.2.2	Conformance of Documents	53
15.2.3	Conformance of Documentation	53
15.3	Conforming SGML System	53
15.3.1	Conformance of Documentation	54
15.3.2	Conformance to System Declaration	54
15.3.3	Support for Reference Concrete Syntax	54
15.3.4	Support for Reference Capacity Set	55
15.3.5	Consistency of Parsing	55
15.3.6	Application Conventions	55
15.4	Validating SGML Parser	56
15.4.1	Error Recognition	56
15.4.2	Identification of SGML Messages	56
15.4.3	Content of SGML Messages	56
15.5	Documentation Requirements	56
15.5.1	Standard Identification	56
15.5.2	Identification of SGML Constructs	56
15.5.3	Terminology	57
15.5.4	Variant Concrete Syntax	57

iTeh STANDARD PREVIEW
(standards.iteh.ai)

<https://standards.iteh.ai/catalog/standards/sist/bb725189-cdc0-4cac-8cc4-06c1b29545ad/iso-8879-1986>

15.6	System Declaration	57
15.6.1	Concrete Syntaxes Supported	57
15.6.1.1	Concrete Syntax Changes	57
15.6.1.2	Character Set Translation	57
15.6.2	Validation Services	58

Annexes

A	Introduction to Generalized Markup	59
A.1	The Markup Process	59
A.2	Descriptive Markup	60
A.3	Rigorous Markup	62
A.4	Conclusion	64
A.5	Acknowledgments	65
A.6	Bibliography	65
B	Basic Concepts	66
B.1	Documents, Document Type Definitions, and Procedures	66
B.1.1	Documents	66
B.1.2	Document Type Definitions	66
B.1.3	Procedures	67
B.2	Markup	67
B.3	Distinguishing Markup from Text	68
B.3.1	Descriptive Markup Tags	68
B.3.2	Other Markup	69
B.3.3	Record Boundaries	69
B.3.3.1	Record Boundaries in Data	70
B.3.3.2	Record Boundaries in Markup	70
B.4	Document Structure	70
B.4.1	Document Type Definitions	70
B.4.2	Element Declarations	71
B.4.2.1	Content Models	71
B.4.2.2	Connectors and Occurrence Indicators	71
B.4.2.3	Entity References in Models	72
B.4.2.4	Name Groups	72
B.4.2.5	Data Characters	73
B.4.2.6	Empty Content	73
B.4.2.7	Non-SGML Data	73
B.4.2.8	Summary of Model Delimiters	74
B.5	Attributes	74
B.5.1	Specifying Attributes	74
B.5.1.1	Names	74
B.5.1.2	Attribute Values	75
B.5.2	Declaring Attributes	75
B.5.2.1	Attribute Definition Syntax	75
B.5.2.2	Complex Attribute Values	76
B.5.2.3	Name Token Groups	77
B.5.2.4	Changing Default Values	77
B.6	Entities	78
B.6.1	Entity Reference Syntax	78
B.6.2	Declaring Entities	78
B.6.2.1	Processing Instructions	79
B.6.2.2	Entities with Entity References	79
B.6.2.3	External Entities	79
B.6.2.4	Public Entities	80
B.7	Characters	80

B.7.1	Character Classification	80
B.7.2	Character References	81
B.7.3	Using Delimiter Characters as Data	82
B.8	Marked Sections	83
B.8.1	Ignoring a Marked Section	83
B.8.2	Versions of a Single Document	84
B.8.3	Unparsable Sections	84
B.8.4	Temporary Sections	85
B.8.5	Keyword Specification	85
B.8.6	Defining a Marked Section as an Entity	85
B.9	Unique Identifier Attributes	86
B.10	Content Reference Attributes	86
B.11	Content Model Exceptions	87
B.11.1	Included Elements	87
B.11.2	Excluded Elements	87
B.12	Document Type Declaration	88
B.13	Data Content	88
B.13.1	Data Content Representations	89
B.13.1.1	Character Data (PCDATA, CDATA, and RCDATA)	89
B.13.1.2	Non-SGML Data (NDATA)	89
B.13.2	Data Content Notations	90
B.13.2.1	Notations for Character Data	90
B.13.2.2	Notations for Non-SGML Data	91
B.13.2.3	Specifying Data Content Notations	91
B.14	Customizing	92
B.14.1	The SGML Declaration	92
B.14.1.1	Optional Features	92
B.14.1.2	Variant Concrete Syntax	92
B.14.2	Impact of Customization	92
B.15	Conformance	93
C	Additional Concepts	94
C.1	Markup Minimization Features	94
C.1.1	SHORTTAG: Tags With Omitted Markup	94
C.1.1.1	Unclosed Short Tags	95
C.1.1.2	Empty Tags	95
C.1.1.3	Attribute Minimization	95
C.1.2	OMITTAG: Tags May be Omitted	96
C.1.2.1	Tag Omission Concepts	97
C.1.2.2	Specifying Minimization	97
C.1.2.3	End-tag Omission: Intruding Start-tag	98
C.1.2.4	End-tag Omission: End-tag of Containing Element	98
C.1.2.5	Start-tag Omission: Contextually Required Element	99
C.1.2.6	Combination with Short Tag Minimization	99
C.1.2.7	Markup Minimization Considerations	99
C.1.3	SHORTREF: Short Reference Delimiters May Replace Complete Entity References	100
C.1.3.1	Typewriter Keyboarding: Generalized WYSIWYG	100
C.1.3.2	Typewriter Keyboarding Example: Defining a Short Reference Map	100
C.1.3.3	Typewriter Keyboarding Example: Activating a Short Reference Map	101
C.1.3.4	Tabular Matter Example	102
C.1.3.5	Special Requirements	103
C.1.4	DATATAG: Data May Also be a Tag	103
C.1.5	RANK: Ranks May be Omitted from Tags	106
C.2	LINK Features: SIMPLE, IMPLICIT, and EXPLICIT	107
C.2.1	Link Process Definitions	108
C.3	Other Features	108

ISO 8879:1986

<https://standards.iteh.ai/catalog/standards/sist/bbf25d89-cdc0-4eac-8ec4-00cfb29943ad/iso-8879-1986>

C.3.1	CONCUR: Document Instances May Occur Concurrently	108
C.3.2	SUBDOC: Nested Subdocument Entities May Occur	109
C.3.3	FORMAL: Public Identifiers are Formal	109
D	Public Text	110
D.1	Element Sets	110
D.1.1	Common Element Types	110
D.1.2	Pro Forma Element Types	110
D.2	Data Content Notations	110
D.3	Variant Concrete Syntaxes	111
D.3.1	Multicode Concrete Syntaxes	111
D.4	Entity Sets	111
D.4.1	General Considerations	112
D.4.1.1	Format of Declarations	112
D.4.1.2	Corresponding Display Entity Sets	113
D.4.1.3	Entity Names	113
D.4.1.4	Organization of Entity Sets	114
D.4.2	Alphabetic Characters	114
D.4.2.1	Latin	114
D.4.2.2	Greek Alphabetic Characters	117
D.4.2.3	Cyrillic Alphabetic Characters	119
D.4.3	General Use	121
D.4.3.1	Numeric and Special Graphic Characters	121
D.4.3.2	Diacritical Mark Characters	123
D.4.3.3	Publishing Characters	123
D.4.3.4	Box and Line Drawing Characters	125
D.4.4	Technical Use	126
D.4.4.1	General	126
D.4.4.2	Greek Symbols	128
D.4.4.3	Alternative Greek Symbols	129
D.4.5	Additional Mathematical Symbols	130
D.4.5.1	Ordinary Symbols	130
D.4.5.2	Binary and Large Operators	130
D.4.5.3	Relations	131
D.4.5.4	Negated Relations	133
D.4.5.5	Arrow Relations	134
D.4.5.6	Opening and Closing Delimiters	135
E	Application Examples	136
E.1	Document Type Definition	136
E.2	Computer Graphics Metafile	140
E.3	Device-Independent Code Extension	140
E.3.1	Code Extension Facilities	140
E.3.1.1	Avoiding False Delimiter Recognition	141
E.3.1.2	Eliminating Device and Code Dependencies	143
F	Implementation Considerations	145
F.1	A Model of SGML Parsing	145
F.1.1	Physical Input	145
F.1.1.1	Entities	145
F.1.1.2	Record Boundaries	145
F.1.2	Recognition Modes	145
F.1.3	Markup Minimization	146
F.1.4	Translation	147
F.1.5	Command Language Analogy	147

F.2	Initialization	147
F.2.1	Initial Procedure Mapping	147
F.2.2	Link Process Specification	147
F.2.3	Concurrent Document Instances	147
F.3	Dynamic Procedure Mapping	148
F.4	Error Handling	148
G	Conformance Classification and Certification	149
G.1	Classification Code	149
G.1.1	Feature Code	149
G.1.2	Validation Code	150
G.1.3	Syntax Code	151
G.2	Certification Considerations	151
H	Theoretical Basis for the SGML Content Model	152
H.1	Model Group Notation	152
H.2	Application of Automata Theory	152
H.3	Divergence from Automata Theory	153
I	Nonconforming Variations	154
I.1	Fixed-length Generic Identifiers	154
I.2	Single Delimiter	154

ITh STANDARD PREVIEW
(standards.iteh.ai)

Figures

ISO 8879:1986

<https://standards.iteh.ai/catalog/standards/sist/bbf25d89-cdc0-4eac-8ec4-69cd329043e1/iso-8879-1986>

1	Character Classes: Abstract Syntax	29
2	Character Classes: Concrete Syntax	30
3	Reference Delimiter Set: General	31
4	Reference Delimiter Set: Short References	33
5	Reference Capacity Set	49
6	Reference Quantity Set	52
7	Reference Concrete Syntax	54
8	Typical SGML Declaration for Basic SGML Document	55
9	Element Markup	69
10	Start-tag with 2 Attributes	75
11	Multicode Basic Concrete Syntax	112
12	Graphics Metafile Attributes (1 of 2): Encoding and View	141
13	Graphics Metafile Attributes (2 of 2): Size and Rotation	142
14	Function Characters for Device-Independent Multicode Concrete Syntaxes	143

15	FSV Conformance Classification	150
----	--------------------------------------	-----

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO 8879:1986

<https://standards.iteh.ai/catalog/standards/sist/bbf25d89-cdc0-4eac-8ec4-00cfb29943ad/iso-8879-1986>

iTeh STANDARD PREVIEW
(standards.iteh.ai)

This page intentionally left blank

ISO 8879:1986

<https://standards.iteh.ai/catalog/standards/sist/bbf25d89-cdc0-4eac-8ec4-00cfb29943ad/iso-8879-1986>

Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML)

0 Introduction

This International Standard specifies a language for document representation referred to as the "Standard Generalized Markup Language" (SGML). SGML can be used for publishing in its broadest definition, ranging from single medium conventional publishing to multi-media data base publishing. SGML can also be used in office document processing when the benefits of human readability and interchange with publishing systems are required.

0.1 Background

A document can be viewed in the abstract as a structure of various types of element. An author organizes a book into chapters that contain paragraphs, for example, and figures that contain figure captions. An editor organizes a magazine into articles that contain paragraphs that contain words, and so on.

Processors treat these elements in different ways. A formatting program might print headings in a prominent type face, leave space between paragraphs, and otherwise visually convey the structure and other attributes to the reader. An information retrieval system would perhaps assign extra significance to words in a heading when creating its dictionary.

Although this connection between a document's attributes and its processing now seems obvious, it tended to be obscured by early text processing methods. In the days before automated typesetting, an editor would "mark up" a manuscript with the specific processing instructions that would create the desired format when executed by a compositor. Any connection between the instructions and the document's structure was purely in the editor's head.

Early computerized systems continued this approach by adding the process-specific "markup" to the machine-readable document file. The markup still consisted of specific processing instructions, but now they were in the language of a formatting program, rather than a human compositor. The file could not easily be used for a different purpose, or on a different computer system, without changing all the markup.

As users became more sophisticated, and as text processors became more powerful, approaches were developed that alleviated this problem. "Macro calls" (or "format calls") were used to identify points in the document where processing was to occur. The actual processing instructions were kept outside of the document, in "procedures" (or "macro definitions" or "stored formats"), where they could more easily be changed.

While the macro calls could be placed anywhere in a document, users began to recognize that most were placed at the start or end of document elements. It was natural, therefore, to choose names for such macros that were "generic identifiers" of the element types, rather than names that suggested particular processing (for example, "heading" rather than "format-17"), and so the practice of "generic coding" (or "generalized tagging") began.

Generic coding was a major step towards making automated text processing systems reflect the natural relationship between document attributes and processing. The advent of "generalized markup languages" in the early 1970's carried this trend further by providing a formal language base for generic coding. A generalized markup language observes two main principles:

- a) Descriptive markup predominates and is distinguished from processing instructions.

Descriptive markup includes both generic identifiers and other attributes of document elements that motivate processing instructions. The processing instructions, which can be in any language, are normally collected outside of the document in procedures.

As the source file is scanned for markup and the various elements are recognized, the processing system executes the procedures associated with each element and attribute for that process. For other processes, different procedures can be associated with the same elements and attributes without changing the document markup.

When a processing instruction must be entered directly in a document, it is delimited differently from descriptive markup so that it can easily be located and changed for different processes.

- b) Markup is formally defined for each type of document.

A generalized markup language formalizes document markup by incorporating "document type definitions". Type definitions include a specification (like a formal grammar) of which elements and attributes can occur in a document and in what order. With this information it is possible to determine whether the markup for an individual document is correct (that is, complies with the type definition) and also to supply markup that is missing, because it can be inferred unambiguously from other markup that is present.

NOTE — A more detailed introduction to the concepts of generic coding and the Standard Generalized Markup Language can be found in annex A.

0.2 Objectives

The Standard Generalized Markup Language standardizes the application of the generic coding and generalized markup concepts. It provides a coherent and unambiguous syntax for describing whatever a user chooses to identify within a document. The language includes:

- An "abstract syntax" for descriptive markup of document elements.
- A "reference concrete syntax" that binds the abstract syntax to particular delimiter characters and quantities. Users can define

alternative concrete syntaxes to meet their requirements.

- Markup declarations that allow the user to define a specific vocabulary of generic identifiers and attributes for different document types.
- Provision for arbitrary data content. In generalized markup, "data" is anything that is not defined by the markup language. This can include specialized "data content notations" that require interpretation different from general text: formulas, images, non-Latin alphabets, previously formatted text, or graphics.
- Entity references: a non-system-specific technique for referring to content located outside the mainstream of the document, such as separately-written chapters, pi characters, photographs, etc.
- Special delimiters for processing instructions to distinguish them from descriptive markup. Processing instructions can be entered when needed for situations that cannot be handled by the procedures, but they can easily be found and modified later when a document is sent to a different processing system.

For a generalized markup language to be an acceptable standard, however, requires more than just providing the required functional capabilities. The language must have metalinguistic properties, in order to satisfy the constraints imposed by the need to use it in a multiplicity of environments. The major constraints, and the means by which the Standard Generalized Markup Language addresses them, can be summarized as follows:

- a) Documents "marked up" with the language must be processable by a wide range of text processing and word processing systems.

The full form of the language, with all optional features, offers generality and flexibility that can be exploited by sophisticated systems; less powerful systems need not support the features. To facilitate interchange between dissimilar systems, an "SGML declaration" describes any markup features or concrete syntax variations used in a document.

- b) The millions of existing text entry devices must be supported.

SGML documents, with the reference concrete syntax, can easily be keyboarded and understood by humans, without machine assistance. As a result:

- Use of SGML need not await the development and acceptance of a new generation of hardware — just software to process the documents on existing machines.
- Migration to such a new generation (when it comes) will be easier, as users will already be familiar with SGML.

- c) There must be no character set dependency, as documents might be keyed on a variety of devices.

The language has no dependency on a particular character set. Any character set that has bit combinations for letters, numerals, space, and delimiters is acceptable.

- d) There must be no processing, system, or device dependencies.

Generalized markup is predominantly descriptive and therefore inherently free of such dependencies. The occasional processing instruction is specially delimited so it can be found and converted for interchange, or when a different process renders the instruction irrelevant.

References to external parts of a document are indirect. The mappings to real system storage are made in “external entity declarations” that occur at the start of the document, where they can easily be modified for interchange.

The concrete syntax can be changed with the SGML declaration to accommodate any reserved system characters.

- e) There must be no national language bias.

The characters used for names can be augmented by any special national characters. Generic identifiers, attribute names, and other names used in descriptive markup are defined by the user in element and entity declarations.

The declaration names and keywords used in markup declarations can also be changed.

Multiple character repertoires, as used in multi-lingual documents, are supported.

- f) The language must accommodate familiar typewriter and word processor conventions.

The “short reference” and “data tag” capabilities support typewriter text entry conventions. Normal text containing paragraphs and quotations is interpretable as SGML although it is keyable with no visible markup.

- g) The language must not depend on a particular data stream or physical file organization.

The markup language has a virtual storage model in which documents consist of one or more storage entities, each of which is a sequence of characters. All real file access is handled by the processing system, which can decide whether the character sequence should be viewed as continuous, or whether it should reflect physical record boundaries.

- h) “Marked up” text must coexist with other data.

A processing system can allow text that conforms to this International Standard to occur in a data stream with other material, as long as the system can locate the start and end of the conforming text.

Similarly, a system can allow data content not defined by SGML to occur logically within a conforming document. The occurrence of such data is indicated by markup declarations to facilitate interchange.

- i) The markup must be usable by both humans and programs.

The Standard Generalized Markup Language is intended as a suitable interface for keyboarding and interchange without preprocessors. It allows extensive tailoring to accommodate user preferences in text entry conventions and the requirements of a variety of keyboards and displays.

However, it is recognized that many implementers will want to take advantage of the language’s information capture capabilities to provide intelligent editing or to create SGML documents from a word processing front-end environment. SGML accommodates such uses by providing the following capabilities:

- Element content can be stored separately from the markup.
- Control characters can be used as delimiters.
- Mixed modes of data representation are permitted in a document.
- Multiple concurrent logical and layout structures are supported.

0.3 Organization

The organization of this International Standard is as follows:

- a) The physical organization of an SGML document as an entity structure is specified in clause 6.