

Implementation of ISO/IEC 10744 : 1992

**Information technology —
Hypermedia/Time-based
Structuring Language
(HyTime)**

National foreword

This British Standard reproduces verbatim ISO/IEC 10744 : 1992 and implements it as the UK national standard.

This British Standard is published under the direction of the Information Systems Technology Standards Policy Committee whose Technical Committee IST/18 has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international committee any enquiries on interpretation, or proposals for change, and keep UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

NOTE. International and European Standards, as well as overseas standards, are available from BSI Sales Department, BSI, Linford Wood, Milton Keynes, MK14 6LE.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

INTERNATIONAL STANDARD

ISO/IEC 10744

First edition
1992-11-01

Information technology — Hypermedia/Time-based Structuring Language (HyTime)

*Technologie de l'information — Langage de structuration
temporelle/hypermédia (HyTime)*



Reference number
ISO/IEC 10744:1992(E)

AVA Documents
Document Sharing Hub

Contents

	Page
1 Scope	1
1.1 Definition of scope	1
1.2 Field of application	2
2 Normative references	2
3 Definitions	2
4 Symbols and abbreviations	7
5 Notation	7
5.1 Architectural forms	8
5.2 Editorial conventions	8
5.3 HyTime SGML conventions	8
5.3.1 Element type forms	9
5.3.1.1 Element type declaration	9
5.3.1.2 Attribute list declaration	10
5.3.2 Attribute list forms	10
5.3.3 Attribute list conventions	10
5.3.3.1 Attribute name	10
5.3.3.2 Declared value prescription	10
5.3.3.3 Default value prescription	11
6 Base module	12
6.1 Concepts and definitions	12
6.1.1 Object representation	12
6.1.1.1 Entity structure	13
6.1.1.2 Data	13
6.1.2 Object identification and addressing	13
6.1.2.1 Name space addressing	13

© ISO/IEC 1992

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

6.1.2.2	Coordinate addressing	14
6.1.2.3	Semantic addressing	14
6.2	Hyperdocument management facilities	14
6.2.1	Object representation	14
6.2.2	Object identification and addressing	15
6.2.3	Object access	15
6.2.4	Hyperdocument interchange format	15
6.2.4.1	Bounded object set (BOS)	15
6.2.4.2	SDIF packer	16
6.2.4.3	SDIF unpacker	16
6.3	HyTime identification facilities	16
6.3.1	APPINFO parameter of SGML declaration	16
6.3.2	HyTime support declarations	17
6.3.2.1	HyTime version declaration	17
6.3.2.2	HyTime module declarations	17
6.4	HyTime document	18
6.4.1	HyTime/non-HyTime bridge element	20
6.4.2	Non-HyTime element	20
6.5	Common attributes	21
6.5.1	Identification attributes	21
6.5.2	Lexical types	22
6.5.2.1	Lexical model	23
6.5.2.2	Lexicographic ordering definition	24
6.5.3	ID reference resolution control	24
6.5.4	Bit combination skip count	27
6.5.5	Default value list	27
6.5.5.1	Default value list attributes	27
6.5.5.2	Default value list element	28
6.5.6	Descriptive text	28
6.5.6.1	Descriptive text attributes	29
6.5.6.2	Description table	29

6.5.6.3	Descriptive text	30
6.5.6.4	Descriptive text definition	30
6.5.7	Activity tracking policy	30
6.6	HyTime data attributes	32
6.6.1	Container notation form	32
6.6.2	Common data attributes	32
6.7	Property sets	34
6.7.1	Property definition	35
6.7.2	Qualified property name	36
6.7.3	Qualifying name attributes	37
6.8	Summary of base module options	38
7	Measurement module	38
7.1	Coordinate addressing	38
7.1.1	Concepts and definitions	38
7.1.2	Dimension specification	39
7.1.2.1	Overrun handling	40
7.1.2.2	Axis marker list	40
7.1.2.3	Dimension specification list	41
7.2	Scheduling and extents	41
7.2.1	Concepts and definitions	41
7.2.2	Extent specification	41
7.2.3	Scheduled extent	42
7.2.4	Scheduled extent list	42
7.3	Measurement units	43
7.3.1	Measurement domain definition	43
7.3.2	Useful measurement domains	44
7.3.3	Schedule measurement	44
7.4	Dimension referencing	45
7.4.1	Dimension reference facilities	45
7.4.1.1	Implicit dimension reference	45
7.4.1.2	Explicit dimension reference	45

7.4.1.3	Marker function	47
7.4.2	Dimension reference constraints	48
7.4.2.1	Reasonable and computable rule	48
7.4.2.2	Same domain/common granule rule	48
7.4.2.3	Measurement Domain Unit (MDU) rule	49
7.4.2.4	Forced projection rule	49
7.5	Summary of measurement module options	50
8	Location address module	50
8.1	Concepts and definitions	50
8.1.1	Location types	50
8.1.2	Location views	51
8.2	Attribute list architectural forms	52
8.2.1	Location source	52
8.2.2	Multiple location	52
8.2.3	Span location	53
8.3	Name space locations	54
8.3.1	Identified local element	54
8.3.2	Named object	54
8.3.2.1	Named location address	55
8.3.2.2	Name list specification	55
8.4	Coordinate locations	56
8.4.1	Data location address	57
8.4.2	Node locations	58
8.4.2.1	Node lists	59
8.4.2.2	SGML node types and their properties	59
8.4.2.3	Tree combination	60
8.4.2.4	Tree location address	60
8.4.2.5	Path location address	61
8.4.2.6	List location address	62
8.4.2.7	Relative location address	63
8.5	Semantic locations	64

8.5.1	Property location address	64
8.5.2	Notation-specific location address	64
8.5.3	Bibliographic location address	65
8.6	Location ladders	66
8.7	Querying	66
8.7.1	Query attributes	67
8.7.2	Name list query	68
8.7.3	Axis marker query	68
8.8	Summary of location address module options	69
9	Hyperlinks module	69
9.1	Concepts and definitions	69
9.1.1	Object representation	70
9.1.2	Object identification and addressing	70
9.1.3	Link creation	70
9.1.4	Link traversal	71
9.1.5	Object access	72
9.2	Hyperlink architectural forms	72
9.2.1	Independent link	72
9.2.2	Contextual link	74
9.3	Summary of hyperlinks module options	75
10	Scheduling module	75
10.1	Concepts and definitions	75
10.2	Finite coordinate space	76
10.3	Schedules	77
10.4	Event schedule	77
10.4.1	Event group	78
10.4.2	Event	79
10.4.3	Pulses	80
10.5	Objects	80
10.5.1	Object types for multimedia	80
10.5.2	Malleable objects	81

10.5.3	Accessed anchor list	82
10.6	Extent reconciliation	83
10.7	Finite coordinate space location address	85
10.8	Calendar	86
10.8.1	Calendar specification	86
10.8.1.1	Date and Julian date	87
10.8.1.2	Absolute time	87
10.8.2	Dimension references to calendars	88
10.8.2.1	Calendar reference	88
10.8.2.2	Calendar event reference	89
10.9	Summary of scheduling module options	89
11	Rendition module	89
11.1	Object Modification	89
11.1.1	Wand and modifier rules	90
11.1.2	Wand	90
11.1.3	Modifier scope	91
11.1.3.1	Object modifier	91
11.1.4	Patches	91
11.2	Event projection	92
11.2.1	Concepts and definitions	92
11.2.2	Baton rule	93
11.2.3	Baton	94
11.2.3.1	Projector scope	95
11.2.3.2	Event projector	95
11.2.3.3	Projection function	95
11.2.3.4	Scaling reference	96
11.2.4	Indigenous event	96
11.3	Rendition rule	96
11.4	Summary of rendition module options	97
12	Conformance	97
12.1	Conforming HyTime document	97

12.1.1	Basic hyperlinking HyTime document	97
12.1.2	Basic scheduling HyTime document	97
12.1.3	Minimal HyTime document	98
12.1.4	Minimal hyperlinking HyTime document	98
12.1.5	Minimal scheduling HyTime document	98
12.2	Conforming HyTime application	98
12.2.1	Application conventions	98
12.2.2	Conformance of documents	98
12.2.3	Conformance of documentation	98
12.3	Conforming HyTime system	99
12.3.1	Conformance of documentation	99
12.3.2	Conformance to HyTime system declaration	99
12.3.3	Support for minimum HyTime documents	99
12.3.4	Application conventions	99
12.4	Validating HyTime engine	99
12.4.1	Error recognition	99
12.4.2	Identification of HyTime messages	100
12.4.3	Content of HyTime messages	100
12.5	Documentation requirements	100
12.5.1	Standard identification	100
12.5.2	Identification of HyTime constructs	100
12.5.3	Terminology	101
12.6	HyTime system declaration	101

Annexes

A	Useful element types, notations, and instances	102
A.1	HyLex: HyTime lexical model notation	102
A.1.1	HyLex syntax and semantics	102
A.1.1.1	Formal structure	102
A.1.1.2	Delimiters	103
A.1.1.3	Names	103
A.1.1.4	Wildcards	103

A.1.1.5	Hit boundary constraints	103
A.1.1.6	Alternative single-character patterns (ASCP)	104
A.1.1.7	Matching rules	104
A.1.2	HyLex element type	104
A.1.3	Useful HyLex types	105
A.2	HyTime property definition	106
A.2.1	Useful property sets	107
A.2.1.1	SGML lexical properties	107
A.2.1.1.1	HTlex1: SGML character classes	107
A.2.1.1.2	HTlex2: SGML delimiter role names	108
A.2.1.2	SGML and HyTime semantic properties	109
A.2.1.2.1	HTsem1: SGML basic constructs	109
A.2.1.2.2	HTsem2: HyTime constructs	110
A.2.1.2.3	HTsem3: Unparsed SGML text	111
A.3	HyQ: HyTime query notation	111
A.3.1	HyQ semantics	111
A.3.2	HyQ formal definition	112
A.3.3	HyQ element type	116
A.3.4	HyQ examples	116
A.4	Marker functions	117
A.4.1	HyOp: HyTime single operator marker function	117
A.4.2	HyFunk: HyTime function notation	118
A.4.2.1	HyFunk syntax and semantics	118
A.4.2.2	HyFunk element type	119
A.4.2.3	HyTime Measurement Unit ratio (HMURatio)	120
A.5	Measurement	120
A.5.1	Useful Standard Measurement Units	120
A.5.2	Useful measurement domain definitions	121
A.5.2.1	Generic quantum	121
A.5.2.2	Virtual time	121
A.5.2.3	Virtual space	121

A.5.2.4	Systeme International second	121
A.5.2.5	Systeme International meter	122
A.5.3	Other standard measurement units	123
B	Supplementary materials	125

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.

International Standard ISO/IEC 10744 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Annex A forms an integral part of this International Standard. Annex B is for information only.

Introduction

The Hypermedia/Time-based Structuring Language (HyTime), defined in this International Standard, provides facilities for representing static and dynamic information that is processed and interchanged by hypertext and multimedia applications. HyTime is an application of ISO 8879, the Standard Generalized Markup Language (SGML).

HyTime supports the classic bibliographic model of information referencing, whereby it is possible to represent links to anything, anywhere, at any time. The extension of this model to the computerized information age, known as “integrated open hypermedia” (IOH), is the field of application of HyTime.

HyTime provides standardized mechanisms for specifying interconnections (hyperlinks) within and between documents and other information objects, and for scheduling multimedia information in time and space.

Without HyTime, such information is typically embedded in the processing instructions of hypermedia “scripts” that govern the rendition of such documents, and is therefore not usable for other forms of processing. When HyTime is used, those properties of the information that are independent of specific processing are available for processing by applications and platforms other than the one on which the information was created.

It is for the application designer and user to decide which properties can be isolated from the scripts in this way. In an ideal world, the sole consideration would be whether the properties are intrinsic to the information, regardless of how it is processed. For example, the title of this clause is intrinsic information; the font that it appears in normally is not.

In the real world, representation strategies will vary from one situation to another and will depend on such other considerations as the expected uses of the information, the flexibility of the scripting language, and performance considerations. For this reason, HyTime is highly modularized so that application designers need use only the facilities for the properties they care to describe in a standardized way.

HyTime's rules for the standardized expression of hypermedia structuring information are called “architectural forms”. They govern the use of “attributes”, which are SGML representations of the properties of information objects.

0.1 HyTime modules

The architectural forms and attributes of the HyTime language are grouped into six modules, each of which can have both required facilities and options. Support for the modules and their options is indicated by “HyTime support declarations.”

— Base module

The base module consists of independent utility facilities, some of which are optional. The required facilities support hyperdocument management (using SGML) and identification of HyTime properties. The optional facilities provide for lexical definitions of attribute values and character content, strong typing and resolution range control of ID references, optimizing direct access to parts of documents, context-sensitive control of attribute defaulting, lookup tables for commonly used elements, activity tracking, storing multiple entities in a single "container" entity, and defining properties of information objects.

— Measurement module

The facilities of the measurement module are required by the coordinate address facility of the location address module and by the finite coordinate space and rendition modules. They provide the ability to specify the position and extent of objects in various application-defined measurement units.

— Location address module

The location address module allows the identification of objects that cannot be addressed by SGML unique identifiers, and objects that are in external documents.

Three basic types of address may be supported: name, semantic location, and coordinate location. Addressing of aggregate locations is also possible.

If the scheduling module is supported, it is also possible to address locations in objects that are in notations unknown to the application creating the location address.

HyTime's system-independent way of expressing addresses of hypermedia objects also provides the basis of its hyperlinking and scheduling power.

— Hyperlinks module

This module allows connections ("hyperlinks") to be made among objects, either within a single document or among the constituent documents and information objects of a hyperdocument.

— Scheduling module

This module allows events — occurrences of objects — to be scheduled on the coordinate axes of "finite coordinate spaces" in such a way that their positions can be expressed in terms of their relationships to one another. Measurement along the coordinate axes can be in terms of spatial or temporal units.

— Rendition module

When the scheduling module is used, object modification and/or event projection can be used to represent parameters governing the rendition process.

- Object modification

The object modification facility allows specification of the order in which objects are to be modified during rendition and of the "object modifiers" (such as amplifiers and filters) that will affect them.

NOTE 1 The semantics of the modifiers are not defined by HyTime, only the sequence in which they are to be applied.

- Event projection

Rendition requires the projection of events into a coordinate space where they can be perceived; for example, from a coordinate space with a virtual time axis to one based on real time. The event projection facility allows specification of the factors for computing the positions and extents of events in the target coordinate space.

In situations where the rendered position and extent of an event is not predictable (as when user interaction will affect it), the virtual dimensions of the original events may be projected onto real space/time via a formula in some arbitrary user-defined expression language. Such an expression may, among other things, accept late-binding values during rendition to resolve the positions and extents of projected events.

NOTE 2 The semantics of formatting the objects to fit the new extents is not defined by HyTime.

Applications can choose to include rendition information as an essential part of a hyperdocument, or it can be incorporated in the "style sheets" of the processing programs. The choice depends on the nature of the information being rendered. In multimedia documents, for example, rendition style tends to be more essential to the document than is the case in conventional documents.

0.2 HyTime applications

HyTime provides a generic level of support for a variety of applications, rather than the semantics for a specific application (that is, HyTime is like a carrier or infrastructure).

The boundary between an application and HyTime is variable, and is determined by the application designer, who is free to decide how much of the information will be expressed in a standardized way using HyTime and how much will be application-specific (for example, in a data content notation).

Because the semantics of HyTime's architectural forms and attributes are standardized, it will be possible to implement supporting software and/or hardware usable for a variety of applications. Applications can define additional attributes when defining an element type that is based on an architectural form. The semantics of the application-defined element types and attributes are the only ones defined by the applications themselves. They could be standardized by an industry group or formally by a national or international standards body.

HyTime attributes have no intrinsic meaning other than that specified in this International Standard. However, an application can impute additional semantics to them, either implicitly, or by defining appropriate element types and attributes. For example, to HyTime, the "dimension reference" architectural form means only that the dimension of one element is calculated from the dimension of another. An application, however, could specify (if it wished) that use of dimension referencing implies a synchronization relationship between the elements, and could emphasize this by using "sync" as the generic identifier of a dimension reference element type.

HyTime elements can occur wherever an application's DTD and the HyTime meta-DTD allow. A finite coordinate space could occur, for example, within a paragraph of a memo in order to represent a calendar or project plan in that context. By contrast, several paragraphs could occur as the content of a timed event.

Clients of HyTime, including applications and application architectures, can define non-HyTime architectural forms as well as elements. Although an application may not add architectural forms to HyTime, nor combine HyTime forms with one another, it can create its own attribute (for example, "MyArch") in order to identify its own set of architectural forms. If, for example, a document contains HyTime and MyArch architectural forms, after the content and attributes of each element are processed and validated in SGML terms by the SGML parser, elements with HyTime attributes would be subject to processing and validation by the HyTime engine, while elements with MyArch attributes would be subject to appropriate processing and validation by the application, perhaps aided by a MyArch engine.

HyTime defines some of the parameters needed by an application to accomplish rendition, and some of the rendition functionality. The remainder is provided by the application, or by a document architecture to which the application conforms.

Many different HyTime-compliant applications and architectures could exist, to address different requirements and serve different user constituencies. Such architectures could be incompatible in their non-HyTime aspects, but would still be supportable by a single HyTime engine.

NOTE 3 For example, no application would need to invent its own system for representing finite coordinate spaces, even if its projection functions were extremely intricate and application-specific. HyTime allows application-specific projection functions, using application-chosen (or defined) function languages, to be represented in conjunction with standardized representations of the unprojected and projected finite coordinate spaces.

HyTime's design is optimized for the sequencing and alignment problems encountered in typical hypermedia applications; it is not intended as a general architectural solution for compound document page layout, for which other solutions are better suited.

NOTE 4 However, HyTime is compatible with a wide variety of such solutions. For example, HyTime finite coordinate spaces could be used to describe the media onto which page description language objects are imaged.

0.3 Organization of this document

The structure of this document reflects the modular structure of HyTime, as follows:

- The base module clause (clause 6) is a prerequisite for the other clauses. Some of the facilities it describes are required for all uses of HyTime.
- The measurement module clause (clause 7) describes facilities that are required for facilities in the location address and scheduling modules.
- The location address (clause 8), hyperlinks (clause 9), and scheduling (clause 10) clauses describe modules that are independent of one another.
- The rendition clause (clause 11) describes a module that is dependent on the scheduling module.
- The conformance clause (clause 12) describes requirements that apply to all conforming HyTime documents, applications, and systems.

The text of this International Standard also includes the following annexes:

— annex A

This normative annex defines some useful facilities, including notations for lexical modeling, querying, and calculating dimensions.

— annex B

This informative annex identifies sources of supplementary tutorial and reference materials for HyTime.

Information technology — Hypermedia/Time-based Structuring Language (HyTime)

1 Scope

1.1 Definition of scope

This International Standard defines a language and underlying model for the representation of “hyperdocuments” that link and synchronize static and dynamic (time-based) information contained in multiple conventional and multimedia documents and information objects. The language is known as the “Hypermedia/Time-based Structuring Language”, or “HyTime”.

HyTime can represent time in both the abstract, or “musical” sense, and in user-defined real-time units. It also provides a way of relating the two so that elements of time-dependent documents can be synchronized.

NOTE 5 This facility extends to the representation of multimedia information the power, once limited to conventional documents, to distinguish intrinsic information content from style considerations.

HyTime's techniques for representing its time model are equally applicable to spatial and other domains; all are treated as systems for measuring along different axes of a coordinate space. Arbitrary cross-references and access paths based on external interactions (“hypermedia links”) are also supported.

HyTime's time representation contains sufficient information to derive the durations of both control (“gestural”) data (e.g., control information for audio or video hardware) and visual data (e.g., a music score, presentation storyboard, or television script).

The media formats and data notations of objects in a HyTime hyperdocument can include formatted and unformatted documents, audio and video segments, still images, and object-oriented graphics, among others. Users can specify the positions and extents of occurrences of objects in space and time, using a variety of measurement units and granularities. Temporal requirements of applications ranging from animation to project management can be supported by choosing appropriate measurement granules.

NOTE 6 This International Standard does not address the representation of audio or video content data, but simply defines the means by which the start-time and duration of such data can be synchronized with other digitized information. Nor does it specify the layout process by which occurrences of unformatted documents and other information objects can be made to fit the positions and extents specified for them.

HyTime is an enabling standard, not an encompassing one. As a result, the objects comprising a HyTime hyperdocument are free to conform to any application architectures, or to document architectures imposed by standards, and to be represented in any notation permitted by those architectures. Only the “hub document”, which determines the hyperdocument membership, must conform to HyTime in addition to any other architectures to which it may conform.

HyTime is designed for flexibility and extensibility. Optional subsets can be implemented, alone or in conjunction with user-defined extensions.

The Hypermedia/Time-based Structuring Language (HyTime) is an SGML application conforming to International Standard ISO 8879 — Standard Generalized Markup Language.

The hyperdocument interchange format recommended in this International Standard is ISO 9069, the SGML Document Interchange Format (SDIF). SDIF is defined in Abstract Syntax Notation 1 (ISO 8824) and can be encoded according to the basic encoding rules of ISO 8825 for interchange using protocols conforming to the Open Systems Interconnection (OSI) model. Other interchange formats can also be used.

1.2 Field of application

The field of application of HyTime is "integrated open hypermedia" (IOH), the "bibliographic model" of hyperlinking wherein an author can, by a suitable reference, link to anything, anywhere, at any time.

Because of HyTime's modular design and flexible conformance rules, implementations need support only those facilities that are within their present capabilities. User investment in hyperdocument preparation is nevertheless encouraged because of the well-defined upward-compatible path to a full hypermedia solution.

HyTime is intended for use as the infrastructure of platform-independent information interchange for hypermedia and synchronized and non-synchronized multimedia applications. Application developers will use HyTime constructs to design their information structures and objects, and the HyTime language to represent them for interchange.

NOTE 7 The HyTime language is not intended for encoding the internal representation of information on which application programs act while executing.

Applications can use HyTime to represent hyperdocuments containing information that is at any stage of rendition, from "revisable" to "optimized for interactive access". An application can also choose to convert a rendition of a HyTime hyperdocument into an optimized form for transmission or interactive presentation.

NOTE 8 Whether the HyTime representation of a hyperdocument can be used in a local file system for direct access by programs will depend on the type of information in the hyperdocument, the speed of the platform, and the functions performed by the applications that access the hyperdocument.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8879:1986, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*.

ISO 9069:1988, *Information processing — SGML support facilities — SGML Document Interchange Format (SDIF)*.

ISO/IEC 9070:1991, *Information technology — SGML support facilities — Registration procedures for public text owner identifiers*.

3 Definitions

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

3.1 anchor: An object (or aggregation of objects) that is linked by a hyperlink.

NOTE 9 The term "object" is not a formal construct in HyTime; an anchor could be a document, an element, an arbitrary chunk of data, or any other thing.

NOTE 10 An object is an anchor if and only if a hyperlink identifies it as such.

NOTE 11 An object could be identified as an anchor by several link ends, in the same or different hyperlinks.

3.2 application BOS: A bounded object set that is determined by an application according to its own rules.

NOTE 12 The rules can provide for automatic determination from a set of parameters, selection by the user, or some combination thereof. One possibility is for the application to use the entities included in a HyTime BOS as a starting point, but to allow the user to add or subtract entities from it.

3.3 bit combination: An ordered collection of bits (for example, a byte is a combination of 7 or 8 bits). A bit combination represents a character in character data or markup, but can represent numeric or other values in non-character data.

3.4 bounded object set; BOS: The subject that a HyTime application processes: a set of one or more documents and other information objects.

NOTE 13 There are two kinds: HyTime BOS and application BOS.

3.5 dimension: Size and position on a coordinate axis. It consists of three components: a position (the first occupied quantum), a quantum count (the total number of quanta occupied), and the last occupied quantum.

3.6 document: A collection of information that is identified as a unit and that is intended for human perception.

3.7 document (type) definition; DTD: Rules, determined by an application, that apply SGML to the markup of documents of a particular type. A document type definition includes a formal specification, expressed in a document type declaration, of the element types, element relationships and attributes, and references that can be represented by markup. It thereby defines the vocabulary of the markup for which SGML defines the syntax.

NOTE 14 A document type definition can also include comments that describe the semantics of elements and attributes, and any application conventions.

[ISO 8879]

3.8 entity descriptor: A component of an SDIF data stream that represents an external entity. [ISO 9069]

3.9 entity tree: A tree structure whose nodes are entities, constructed by the following steps:

- a) An SGML document entity or SGML subdocument entity is selected as the root node.
- b) The set of external entities identified by external identifier parameters of markup declarations in the node comprise the children of the node. Each child that is an SGML document entity or SGML subdocument entity is selected as a parent node.
- c) Step b) is repeated for each parent node, until the leaves of the tree or a specified maximum number of levels are reached.

NOTE 15 The maximum number of levels is an attribute of a HyTime document when used as a hub, and can be overridden when a HyTime application is invoked.

3.10 event: The occurrence of an object in a coordinate space. It associates the object with a scheduled extent.

NOTE 16 The scheduled extent of an event gives the object a position and size. The set of first quanta on all the axes defines the position, while the set of quantum counts for all axes completes the dimension specifications and defines the size.

3.11 event projection: The conversion of the scheduled extent of an event from one coordinate space to another (the "unprojected" to the "projected").

NOTE 17 For example, from music time to real time, or from user device coordinates to real space units.

3.12 external identifier: A parameter of an SGML markup declaration (typically an entity declaration) that identifies an external information object.

NOTE 18 It may do so by various means, including:

- a formal public identifier; that is, a globally unique public identifier, which allows a system to access its object by means of a table look-up; and/or
- a system identifier; that is, a file identifier, storage location, program invocation, data stream position, or other system-specific means of locating the object in storage.

3.13 hub document: The document in which access to a hyperdocument begins. In a HyTime hyperdocument, the hub document also defines a HyTime BOS for interchange, rendition, or other processing.

NOTE 19 “Hub document” is not a permanent state of a document and cannot be specified by means of an attribute. The designation of a hub document is a parameter of processing and is specified when an application is invoked.

NOTE 20 It is possible that several HyTime documents, when designated as a hub document, could define the same HyTime BOS.

3.14 hyperdocument: Two or more documents or other information objects that are connected to one another by a web.

NOTE 21 Access to a hyperdocument begins from a designated hub document.

3.15 hyperlink: An information structure that represents a relationship among two or more objects.

NOTE 22 Objects that are related by a hyperlink are called the “anchors” of the hyperlink. An anchor is identified by a property of the hyperlink called a “link end”.

NOTE 23 Hyperlinks can be assigned link types and names by applications and architectures that use HyTime.

NOTE 24 An SGML document can represent relationships by means other than hyperlinks; for example, the subordinate and sibling relationships of the document hierarchy are represented by the position of markup tags.

3.16 hypermedia application: An information processing application that has hypertext and/or multimedia capabilities.

NOTE 25 An implication of this definition is that “hypermedia” refers to the union of hypertext and multimedia, rather than their intersection.

NOTE 26 While it would be possible to maintain a strict distinction among the terms “hypertext”, “multimedia”, and “hypermedia”, there is generally little point in doing so in this International Standard when referring to documents or applications. Events, however, are referred to as “multimedia” rather than “hypermedia” because their nature is unaffected by whether hyperlinks are made to them. Similarly, hyperlinks may be characterized indiscriminately as “hypertext”, “hypermedia”, or simply “hyper”, because their nature is unaffected by the objects to which they are linked.

3.17 hypermedia document: A document or hyperdocument that is used in a hypermedia application.

NOTE 27 A hyperdocument is almost always a hypermedia document as well, in that it is normally used in a hypermedia application. However, not all hypermedia documents are hyperdocuments; for example, a document containing video clips but no hyperlinks.

3.18 Hypermedia/Time-based Structuring Language; HyTime: A standardized hypermedia structuring language for representing hypertext linking, temporal and spatial event scheduling, and synchronization. HyTime provides basic identification and addressing mechanisms and is independent of object data content notations, hyperlink types, processing and presentation functions, and other application semantics. Hyperlinks can be established to documents that conform to HyTime and to those that do not, regardless of whether those documents can be modified. The full HyTime function supports “integrated open hypermedia” (IOH) — the “bibliographic model” of referencing that allows hyperlinks to anything, anywhere, at any time — but systems need support only the subset that is within their present capabilities.

3.19 HyTime attribute: An attribute whose definition is included in a HyTime architectural form.

3.20 HyTime BOS: A bounded object set that consists either of a single document, or of a hyperdocument whose documents and information objects are stored in the nodes of a single entity tree.

NOTE 28 A HyTime BOS can be determined automatically by a HyTime engine. The root of the entity tree is the SGML document entity of the hub document.

3.21 HyTime document: An SGML document whose properties are represented essentially as defined in this International Standard.

NOTE 29 A HyTime document is normally a conforming HyTime document (see 12.1).

3.22 HyTime element: An instance of a HyTime element type.

3.23 HyTime element type: An element type in a HyTime document that conforms to a HyTime architectural form.

NOTE 30 The element type itself is not defined by HyTime. In addition to attributes defined by the architectural form, it may have application-specific attributes.

3.24 HyTime engine: A program (or portion of a program or a combination of programs) that recognizes HyTime constructs in documents and performs application-independent processing of them.

NOTE 31 For example, a HyTime engine can interface with data base and network servers to resolve and access anchors. It can also perform the calculations to locate events consistently in a coordinate space regardless of the schedules in which they were entered or the measurement units used to define their extents.

3.25 HyTime hyperdocument: A hyperdocument whose hub document is a HyTime document.

NOTE 32 The hub document must be represented in SGML; the other components of the hyperdocument need not be. The hub or any other component may conform to a document architecture.

3.26 HyTime system: An SGML system that includes a HyTime engine.

3.27 hypertext: Information that can be accessed in more than one order.

NOTE 33 A hypertext can be a single document or a library of documents (a "hyperdocument"). For example:

- A novel is typically not a hypertext.
- A book with footnotes or internal cross-references is a single document hypertext.
- A book with external cross-references (e.g., bibliographic citations) is a member of a library that as a whole constitutes a hyperdocument.
- A book with both internal and external cross-references is both a single document hypertext and a member of a library that as a whole constitutes a hyperdocument.

3.28 integrated open hypermedia; IOH: The formalization, for computer processing, of the "bibliographic model" of referencing that allows representation of hyperlinks to anything, anywhere, at any time.

3.29 link: Depending on context, either a hyperlink or an SGML processing link ("link process").

3.30 link end: A property of a hyperlink that identifies an anchor by its location.

NOTE 34 The term "location" is not a formal construct in HyTime; a link end could locate an anchor by a name, a coordinate position, or any semantic construct.

3.31 link process definition; LPD: Application-specific rules that apply SGML to describe a link process. A link process definition includes a formal specification, expressed in a *link type declaration*, of the link between elements of the source and result, including the definitions of source attributes applicable to the link process ("link attributes").

NOTE 35 A link process definition can also include comments that describe the semantics of the process, including the meaning of the link attributes and their effect on the process.

[ISO 8879]

3.32 link type: A class of hyperlinks. It assigns meaning to the relationship represented by the hyperlinks, including the role in the relationship played by each anchor.

3.33 multimedia: (Adjective) Employing more than one means of communicating something, such as forms used by artists, musical composers, etc.

3.34 (object) modification: The modification of an object by another object during rendition.

NOTE 36 For example, routing audio signals through an effects box.

NOTE 37 HyTime deals with the scheduling of modifiers, but not the semantics of modification.

3.35 patch: An interconnection of modifiers and the objects they modify.

3.36 presentation: A state of processing of a document in which it is ready for human perception.

3.37 previous specified element: The element in the SGML representation of the document that was most recently parsed.

NOTE 38 It is typically also the one most recently occurring when the SGML representation is viewed as a character string, although an entity reference could cause other elements to be parsed more recently than that one.

3.38 projection: Event projection.

3.39 quantum: Indivisible, countable division of a coordinate axis.

3.40 real time: Time in the every-day sense, as measured in seconds, minutes, hours, etc.

3.41 rendition: A process performed on a HyTime document to prepare it for presentation. It may include event projection and object modification as defined by HyTime, in addition to application-specific processing.

NOTE 39 Several renditions might be performed in sequence in order to create the presentation that is ultimately perceived by a user. The renditions could involve projecting events to a new coordinate space; for example, from virtual time to real time.

3.42 reportable HyTime error; RHE: A failure of a HyTime document to conform to the requirements of this International Standard other than one that:

- a) cannot be detected without processing the document; or
- b) is identified in this International Standard as not being reportable.

3.43 (scheduled) extent: Size and position in a coordinate space. It consists of a dimension on each axis of the space.

3.44 SDIF packer: A program that creates an SDIF data stream. [ISO 9069]

3.45 SDIF unpacker: A program that decomposes an SDIF data stream into its constituent entities.

NOTE 40 If necessary, the SDIF unpacker will modify the system identifier parameter of markup declarations to be consistent with storage addresses in its environment.

[ISO 9069]

3.46 SGML Document Interchange Format; SDIF: A data structure that enables a main document and its related documents, each of which might be stored in several entities, to be combined into a single data stream for interchange in a manner that will permit the recipient to reconstitute the separate entities.

NOTE 41 When SDIF is used as a hyperdocument interchange format for HyTime, the "main" document, if there is more than one, is the hub document.

[ISO 9069]

3.47 SGML processing link: The "link process" feature of SGML; it allows multiple context-sensitive sets of presentation and processing specifications to be associated with elements of a document.

3.48 SGML subdocument entity; SUBDOC: An SGML entity that conforms to the SGML declaration of the SGML document entity, while conforming to its own document type and link type declarations. It contains, at a minimum, a base document type declaration and the start and end of a base document element. [ISO 8879]

3.49 SMU name: A name, declared by a notation declaration, that identifies a standard measurement unit.

3.50 validating HyTime engine: A conforming HyTime engine that can find and report a reportable HyTime error if (and only if) one exists.

3.51 view: A presentation of a web and the anchors that it connects.

3.52 web: A set of one or more hyperlinks that are used together.

NOTE 42 Typically, the hyperlinks deal with a common topic and/or they can be traversed continuously through common anchors and/or anchors occurring in the same object.

4 Symbols and abbreviations

The following abbreviations are used in this International Standard.

BOS	Bounded object set.
DTD	Document type definition.
FCS	Finite coordinate space.
GMT	Greenwich Mean Time.
HMU	HyTime measurement unit.
HyTime	Hypermedia/Time-based Structuring Language.
IOH	Integrated open hypermedia.
JD	Julian date.
LPD	Link process definition.
MDU	Measurement domain unit.
MJD	Modified Julian date.
QLTN	Qualified lexical type name.
QPN	Qualified property name.
RHE	Reportable HyTime error.
SDIF	SGML Document Interchange Format.
SGML	Standard Generalized Markup Language.
SMU	Standard measurement unit.
SUBDOC	SGML subdocument entity.
UTC	Coordinated Universal Time.

5 Notation

The specification of HyTime is accomplished by a combination of narrative text and formal definitions.¹⁾ The formal definitions are expressed in SGML.²⁾

1) This document makes several references to industry and proprietary standards, products, user groups, and publications. Such references are not normative, and do not imply endorsement by ISO, IEC, or their national member bodies or affiliates. Any brand names or trademarks mentioned are the property of their respective owners.

2) The formal SGML definitions are part of the text of this International Standard and are protected by copyright. In order to facilitate conformance to HyTime, the formal SGML definitions of architectural forms in the body of this International Standard, and of the element types and instances in annex A, may be copied as specified in the following copyright notice: (C) International Organization for Standardization 1992. Permission to copy in any form is granted for use with conforming HyTime systems and applications as defined in ISO/IEC 10744, provided this notice is included in all copies. The permission to copy does not apply to any other material in this International Standard.

NOTE 43 This document uses editorial conventions mandated by the ISO with which the reader should be familiar in order to understand the implications of certain words. Sources for a brief informal explanation of these conventions, and of the subset of SGML that is used in this document, can be found in annex B. However, the specific conventions for using SGML to define HyTime architectural forms are described in this clause.

The text describing each construct emphasizes semantics, while the formal SGML definition provides the rigorous syntactic definitions underlying the text descriptions. The text occasionally discusses syntactic issues that require explanation beyond that provided in the formal SGML, but normally such issues as the syntactic constraints on an attribute value, or the default value supplied by HyTime, are not described in the text.

NOTE 44 For this reason, it is recommended that the reader refer to the SGML definitions while reading the textual descriptions. Although the SGML definition always follows the related text, the user may find it helpful to read the SGML first in some cases.

5.1 Architectural forms

As HyTime is an enabling tool, it does not specify a single "HyTime document type definition" or a "HyTime document architecture". HyTime is defined formally as a set of rules, called "architectural forms", that application designers can apply in their document type definitions (DTDs).

Architectural forms are rules for creating and processing elements (just as document architectures are rules for creating and processing documents). Architectural forms are specified principally by attribute definitions, one of which is the "HyTime architectural form" attribute, whose value identifies the form. However, an architectural form can also include rules for constructing the content models of element types conforming to the form.

There are two kinds of architectural form:

- | | |
|---------------------|--|
| element type form | This is defined by an element type declaration in conjunction with an attribute definition list declaration. |
| attribute list form | This is defined solely by an attribute definition list declaration. Its attribute definitions can be used only with designated element type forms or data content notations. |

An element type that conforms to a HyTime element type form is called a "HyTime element type", even though the element type itself is not defined by HyTime, but by an application. An instance of a HyTime element type is called a "HyTime element".

An application DTD can include both HyTime and non-HyTime element types. The element type architectural forms rigorously specify where the two kinds of element can occur.

HyTime also defines various element types (conforming to element type forms), several useful elements, and related data content notations.

5.2 Editorial conventions

When a construct is first introduced, it is described in the text. If the construct occurs in the formal SGML specification, both the formal SGML name and a full name in English (if it has one) are presented, as follows:

The element type form **full construct name** (*SGMLname*) ...

The attribute list form *SGMLname* ...

The attribute **full construct name** (*SGMLname*) ...

If a construct occurs in several places (as in the case of the same attribute with several element type forms) the description is not repeated.

5.3 HyTime SGML conventions

The syntax and structure of the HyTime architectural forms and element types are specified rigorously using SGML declarations. The declarations include comments, called "conventional comments", that follow conventions established in this sub-clause to specify HyTime constructs and constraints that are enforced by a HyTime engine and an SGML parser. The declarations conform to valid SGML syntax in every respect.

NOTE 45 This technique is used so that the declarations can be copied for use as models when creating HyTime DTDs and will parse properly.

The conventional comments do not extend SGML in any way. They are used in this International Standard only, as a notation for the documentation of HyTime. They need not be included in HyTime application DTDs and, if they are included, the SGML parser will ignore them as it would any other comment.

NOTE 46 A HyTime engine must intrinsically be aware of the information the comments convey. In other words, the comments need to be read by programmers creating HyTime engines, not by the engines themselves.

Declarations that define architectural forms are presented with the declaration names in lower-case. Those that define element types (that is, not element type forms) follow the normal SGML convention; their declaration names are presented in upper-case.

NOTE 47 Note that the names of architectural forms, unlike those of element types, are not necessarily meant to be used in DTDs. Although element type form names can be used when the application requires only one element type that conforms to the form, the names of attribute list forms never are, as they have no counterparts in a DTD.

5.3.1 Element type forms

An element type form is defined by an element type declaration in conjunction with an attribute list declaration.

5.3.1.1 Element type declaration

The structure of an element type form is defined by a "meta" content model in the element type declaration. The model is "meta" in the sense that its "element types" are really architectural forms and mean "any element conforming to that form". The text states whether HyTime associates any semantics with the content. If the text is silent, the elements have no special semantics in their role as content of their containing element. However, they still have their normal HyTime semantics as individual elements.

NOTE 48 For example, no special HyTime semantics are defined for the content of an activity tracking policy element, but a contextual link element occurring in the content would retain its normal HyTime semantics.

The set of declarations defined in this International Standard collectively represent a meta-DTD. It is a reportable HyTime error (RHE) if an element occurs in a HyTime document instance at a point where its architectural form is not permitted by the HyTime meta-DTD, or if one fails to occur at a point where an instance of a particular architectural form is required.

HyTime does not constrain the construction of document type definitions, only document instances. Any document type declaration conforming to SGML is allowed in a HyTime document as long as the document instance set conforms to HyTime.

NOTE 49 A validating HyTime engine can optionally report DTD constructs (such as poorly designed content models) that would prevent the creation of a valid conforming instance, or that would allow the creation of a nonconforming instance.

As with a normal content model, inclusion exceptions can be used to define a meta-model. Use of inclusions in this way means that elements of the named types can occur anywhere in the content in the document instance, including the content of subordinate elements. The application designer can choose, in his DTD, to require the elements to be proper subelements, inclusions, or both.

NOTE 50 An example of a content model with an inclusion is:

```
<!element HyDoc      -- HyTime document element --
- O      (%HyBrid;)* +(%loc;|%link;) >
```

The meta-DTD comprising the SGML definitions in this International Standard includes all of the optional facilities of HyTime. The actual meta-DTD applicable to a given document could therefore vary (see 6.4). However, for any given HyTime system declaration (see 12.6) there will be only one meta-DTD for all HyTime applications. That is, the constraints of a more restricted meta-DTD that is implied by a document's support declarations is enforced by the document's DTD.

NOTE 51 The declared content or content model of an application's element type can differ from that of the architectural form as long as it allows instances that conform to the architectural form meta-model. For example, a meta-model might be

"(alblc)". An application designer could legally exclude the possibility of "c" by using the content model "(alb)" (or even "(alx)", except that if an "x" occurred in the instance it would be an RHE).

Character data content is further specified (beyond the normal SGML tokens) by means of the "HyLex" notation and "reftype" conventional comments, which are described in connection with their use in documenting attribute values in 5.3.3.2.

5.3.1.2 Attribute list declaration

The first attribute in an element type form's attribute definition list is the mandatory HyTime attribute, which identifies the architectural form. The HyTime attribute is unique in that it must be specified for every element that is to receive HyTime-defined processing.

NOTE 52 Normally, every element of the same type has the same architectural form, which allows the value to be fixed in the DTD. However, this is not a requirement: an element type definition, could, for example, allow its instances to be one of a choice of architectural forms.

The other attribute definitions follow the conventions described in 5.3.3.

The specification of an element type form's attribute list also includes common attributes (explained below) when they are required for the element type, or when they are important enough to the element type to warrant the emphasis. For example, the ID attribute is included for a form if there are HyTime elements that must reference instances of the form.

An application's attribute list for an element type can also include attributes from certain attribute list forms. The list form names occur in conventional comments in the SGML definition, along with the word "attributes".

5.3.2 Attribute list forms

An attribute list form defines attributes that can be used in the attribute lists of element types conforming to one or more element type forms, including link attribute lists. The element types with which the attributes can be used are indicated in the following manner:

- If the attribute list form name begins with "all-", the attributes can be used with any HyTime or non-HyTime element type. These attributes are known as "common attributes".
- In all other cases, the attribute list declaration contains a conventional comment incorporating the word "use:", followed by a list of the element type form names with which the attributes can be used.

In addition, two attribute list forms are defined for use with data content notations. They are explained fully in 6.6.

5.3.3 Attribute list conventions

All attributes but the HyTime attribute, whether part of an element type form or an attribute list form, are specified using the conventions in this sub-clause.

5.3.3.1 Attribute name

An SGML attribute definition consists of an attribute name, a declared value prescription, and a default value prescription. The components of the definition can be separated by comments, as can the individual definitions in a declaration.

NOTE 53 For a given element type, an application can replace a HyTime attribute's standard name by using the "HyNames" attribute (see 6.5.1).

5.3.3.2 Declared value prescription

Lexical types of attributes (and data content) defined by HyTime are specified by conventional comments containing the word "lertype" or "ulextype", followed by a lexical model in the "HyTime lexical model notation" (HyLex) (see A.1). The comments occur within the attribute definition or element declaration. The word "lertype" signifies a normalized lexical model, while "ulextype" signifies an unnormalized one.

NOTE 54 HyLex is not only used to document HyTime. It is a facility of HyTime and can therefore be used by application designers as well.

Semantic constraints on the value of an attribute are defined in a conventional comment containing the word "constraints:". Either the comment is self-explanatory, and/or there is sufficient additional information in the accompanying text to explain the constraint.

Declared value prescriptions that require ID references are specified with either the SGML IDREF or IDREFS keywords, or with ID references indicated by a lextype conventional comment. If the attribute definition contains a conventional comment containing the word "reftype" followed by an SGML OR group, the target of the ID references must be an element that conforms to an element type form whose name occurs in the OR group. Note that the OR group is meta: the names are architectural forms rather than element types.

An application DTD can use any declared value prescription whose values could satisfy the declared value prescription in this International Standard.

NOTE 55 This requirement can be satisfied by a declared value prescription that allows ID references that resolve to objects that satisfy the declared value prescription of this International Standard.

5.3.3.3 Default value prescription

HyTime attributes are categorized as being "mandatory" or "non-mandatory", and also as being "fixed" or "variable", as follows:

— Non-mandatory attributes

These attributes may be defined in the application DTD, but need not be. If they are not, HyTime provides a value for them. That value is documented either in the default value prescription of the attribute definition, or in a conventional comment that begins with "Default:".

— Mandatory attributes

These attributes must be defined in the application DTD and values must be provided by the application, either by defaulting or by specification on the start-tag. The mandatory status of an attribute is documented in the default value prescription of the attribute definition by the presence of the keyword "#REQUIRED" or the string "in-DTD", and/or by the absence of both an attribute value and a "Default:" conventional comment.

NOTE 56 The "HyTime" attribute is also a mandatory attribute, although it does not follow the above documentation conventions; it is recognized as such by its name.

NOTE 57 An optional facility can have mandatory attributes; however, the attributes are mandatory only if the facility is supported.

— Fixed attributes

If the default value prescription contains the keyword "#FIXED", it signifies that the attribute is a fixed attribute; that is, the attribute must have the same value for all elements of a given element type.

NOTE 58 An application should use a #FIXED value in the DTD in this case, but it is not required because HyTime only defines conformance for document instances. However, a validating HyTime engine can optionally warn of the absence of the keyword.

— Variable attributes

An attribute that is not a fixed attribute is a variable attribute, indicated by the absence of the "#FIXED" keyword in the default value prescription.

In an application DTD, the default value prescription for a variable attribute, whether mandatory or not, can use any keyword or attribute value specification allowed by SGML. The form used to document the attribute in this International Standard is irrelevant, except that:

- If a non-mandatory attribute is not defined, or a defined impliable attribute has no default value and is not specified on the start-tag, the HyTime engine uses the default value documented in this International Standard.

NOTE 59 An application can conveniently specify the defaulting of impliable attributes by using the default value list facility (see 6.5.5).

- Attributes documented in this International Standard as “#REQUIRED” or “#CURRENT” should normally be defined as such by an application.

NOTE 60 The following list gives examples of the forms of declared value and default value prescriptions used to document each combination of attribute categories:

— Mandatory/fixed

NAMES #FIXED in-DTD

— Mandatory/variable

HyTime	NAME	HyDoc
	NAME	#REQUIRED
	CDATA	#CONREF
	IDREFS	#CURRENT

— Non-mandatory/fixed

NUMBERS	#FIXED "1 1"
	CDATA #FIXED "" -- Default: none --

— Non-mandatory/variable

NUMBER	#IMPLIED	-- Default: not known --
	IDREF	#CONREF -- Default: syntactic content --
	CDATA	#CURRENT -- Default: last specified --
	NAMES	novamp
	(order disorder)	order

6 Base module

This clause describes the HyTime base module, which is required for all uses of HyTime.

6.1 Concepts and definitions

Some key concepts relating to the facilities of the base module are described in this sub-clause.

6.1.1 Object representation

SGML is used for object representation in a HyTime document. This sub-clause describes the key SGML constructs on which HyTime relies.

NOTE 61 Components of a HyTime hyperdocument that are not HyTime documents need not be represented in SGML. A document that is used as a hub document of a HyTime hyperdocument must be a HyTime document, and therefore must be represented in SGML.

SGML is a conceptual tool for the modeling of information structures intended for human perception (called “documents”), as well as a notation for representing them. Documents are analyzed and represented in terms of three main constructs:

element A structural building block that can be defined to contain data, subordinate elements, or both. In typical documents, examples of elements would be paragraphs, chapters, headings, figures, and tables. In hypermedia documents, some examples are hyperlinks of various types, event schedules, events, and objects encoded in a variety of data notations.

attribute A property, associated with elements of a given type, whose value describes the element but is not part of its content. For example, the editor of a chapter or revision date of a table. Many element types

have an *ID* attribute, which provides a label for each element of that type so that it can be referenced explicitly from any place in the document. The references are made by elements that have an *ID reference* attribute whose value is the label of the element referenced (as defined by that element's *ID* attribute).

entity A unit of information storage that contains part of a document (sometimes all of one). An entity can be referenced by a name from one or more places in a document, thereby causing its information to be included in the document at the points of reference (see 6.1.1.1).

Entities are independent of elements. For example, an entity could contain part of a picture element or one and one-half paragraph elements.

The particular instances of those constructs that are permitted in a given document are declared in a "document type definition" (DTD) to which the document conforms. The designer of a DTD can choose to allow parts of it to be extended or modified by the user of a particular document (typically by redefining entities that are referenced in the DTD). This technique provides the flexibility necessary to accommodate a wide variety of applications.

6.1.1.1 Entity structure

SGML includes a virtual storage model called the "entity structure" which allows a user to divide a document arbitrarily for ease of management. The entity structure is "virtual" because the mapping from entities to real storage is implementation-dependent, and there need not be a one-to-one relationship between entities and storage objects.

The entity structure is not typically reflected in the part of the document type definition that defines the structure that the application processing is concerned with — the "element structure". The independence of the entity structure is one of the great strengths of SGML, and is essential for hypertext. For example, it relieves an application designer from the burden of having to predict whether chapters will span multiple storage objects or whether a single storage object will contain multiple chapters.

6.1.1.2 Data

The term "data" is used to distinguish the information in a document that is not part of the document structure. For example, the character text within a paragraph, or the raster information representing a photograph, is data. An external entity containing a document that uses a different document representation from the document referencing it is also data.

NOTE 62 Similarly, a recursive instance of an SGML document or subdocument is data with respect to the current instance, because, as it has its own DTD, it is not parsed in the current SGML parsing context.

Data is sometimes referred to as "content", but this usage is ambiguous because the content of an element is not restricted to data; it could also be subelements exclusively, or mixed subelements and data. Data can also occur outside the content of an element in attribute values or external data entities.

NOTE 63 In SGML, data is also defined as "that which is not markup", but the two definitions are consistent: In an SGML document, information that is considered data because it is not structural will also not be markup, and vice versa.

HyTime provides facilities for addressing all forms of data, either in terms of HyTime constructs, or by interfacing to "location addresses" that understand the data representation (see clause 8).

6.1.2 Object identification and addressing

Hypertext linking and multimedia time synchronization are applications of the same basic function — addressing.

A hyperlink addresses its objects by a name that is unique within some name space. Time synchronization uses coordinate addresses of events on a time axis: the address of event A is expressed in terms of its relationship to the address of event B. Spatial alignment is similar, except that the axes are measured in spatial units, rather than temporal.

6.1.2.1 Name space addressing

HyTime's name space addressing comes from SGML. Names are assignable to application-defined information components, called elements, and to units of storage segmentation of information, called entities. By and large, applications deal only with elements, while storage is managed transparently. Therefore, the fundamental form

of name is one that is unique among the elements of a document, known as a unique identifier (ID). All other forms of name are translated to IDs for referencing.

The fundamental means of specifying an object to reference is called an "ID reference" (IDREF). Like the ID, it is an attribute of an element.

For locations outside the current document, HyTime uses a standardized system of identification, including public and private, local and global, unique identifiers (see ISO 9070).

6.1.2.2 Coordinate addressing

A coordinate system in HyTime is called a "finite coordinate space". It consists of a set of coordinate axes and a system for measuring along them.

Each axis is treated as an ordered set of "quanta". A coordinate address consists of a position (the first quantum) and a specific number of quanta for each of the axes of the coordinate space.

When the scheduling module is supported, occurrences of objects ("events") can be given positions in coordinate spaces. Events can be aligned with one another by defining their positions with reference to the positions of other events.

When the location address module is supported, "location address" elements can be defined that associate an ID with a coordinate address. This type of location address allows references to be made to objects that can be identified only by their position.

NOTE 64 For example, "the third word in the sentence".

It also allows references to arbitrary portions of an object.

NOTE 65 For example, "the second and third characters of the third word in the sentence".

6.1.2.3 Semantic addressing

Any object, in any notation, can be represented in a HyTime hyperdocument. The object as a whole could be an element, and therefore could have an ID. The object could be included in an event, and thereby have a coordinate address.

When the location address module is used, it is also possible to address an object by a semantic construct that is not understood by HyTime. Such addressing is done in a data content notation that is appropriate to the object's own notation. The semantic address is equated to an ID by a location address element, just as is done for coordinate addresses, although the interpretation of the semantic address is performed by the application or a system service, rather than by the HyTime engine.

6.2 Hyperdocument management facilities

Hyperdocument management facilities are required for all uses of HyTime. They include the representation, identification, and accessing of objects, and a hyperdocument interchange format.

6.2.1 Object representation

Representation facilities for HyTime documents are provided by SGML and are defined in ISO 8879. HyTime requires the use of the SGML formal public identifier feature, but does not otherwise constrain the choice of available SGML options and variants.

The document representation facilities include:

- Document type definition capability, including user extensions that can be controlled by the application designer.
- Representation of any architectural structure.
- Demarcation of information objects, including components of documents, such as elements, attributes, and data content portions.

- Ability to include multimedia object formats represented in any content notation.
- Separation of processing specifications from intrinsic information.
- Ability to associate multiple sets of processing specifications (“styles”) with element types and individual elements.
- Independent transparent entity structure (see 6.1.1.1).

NOTE 66 Within a HyTime hyperdocument, documents and information objects other than the hub document need not be HyTime documents and need not be represented in SGML. Documents in a HyTime hyperdocument, including the hub document, may conform to a document architecture.

6.2.2 Object identification and addressing

Object identification facilities within HyTime documents are provided by SGML and are defined in ISO 8879. Identification of external documents and information objects is accomplished with public identifiers conforming to ISO 9070.

NOTE 67 ISO 9070 allows public identifiers to be represented in a variety of formats, including ASN.1 structures and SGML formal public identifiers. The identified objects need not be represented in SGML.

NOTE 68 Forms of public identifier not supported by ISO 9070, such as distributed object references with semantic connotations, can be treated as semantic location addresses (see 8.5.2).

NOTE 69 Object identification is required for name space addressing.

6.2.3 Object access

Access from the identification of an object to its physical storage is provided by the SGML entity manager component of a HyTime system, which might invoke network processes to retrieve the located data from a remote data base. The process is analogous to the role of a reference librarian, who accesses a document mentioned in a citation either by locating it on the library shelves, or by consulting a catalog to find it in another library.

6.2.4 Hyperdocument interchange format

A “hyperdocument interchange format” is a data structure that combines the separate documents and information objects of a hyperdocument into a single “data stream” for interchange. The hyperdocument interchange format recommended for use with HyTime is the SGML Document Interchange Format (SDIF) defined in ISO 9069.

NOTE 70 This International Standard does not constrain the interchange format for renditions of a HyTime hyperdocument, for which an application may choose to use a format optimized for transmission or interactive presentation.

NOTE 71 SDIF is defined in ASN.1 (ISO 8824), which provides compatibility with Open Systems Interconnection (OSI) protocols.

SDIF is independent of the architectures and applications of the information objects that it carries. A single SDIF implementation could therefore be capable of supporting all HyTime applications.

6.2.4.1 Bounded object set (BOS)

A hyperdocument is a set of one or more hypermedia documents and other information objects that are connected to one another by hyperlinks, directly or through a web. Traversal of the hyperlinks begins in a document that is designated as the “hub” document at the time of processing.

In a HyTime hyperdocument, designation of a hub document also allows bounds to be placed around the set of objects that is to be interchanged or processed. Without it, since each linked document could have webs that do not link to the hub document, there is a potential for incorporating vast amounts of unwanted, unrelated (or distantly related) information.

A bounded object set can be determined automatically by the HyTime engine by constructing an “entity tree”, starting with the SGML document entity of the hub document as the root. The entity tree includes external entities declared in the hub document, then external entities declared in those entities, and so on. Only entities declared in “external identifier” parameters of markup declarations will be recognized. Therefore, if external

references are made from data content using a notation-specific form of reference, the entities will not be included in the HyTime bounded object set unless the document originator also creates entity declarations for them.

A limit can be placed on the depth of the entity tree by the "bounding level" attribute of the hub document, or by a parameter specified when processing is invoked. The parameter can also specify that the bounded object set is to be determined by the application, rather than by the HyTime engine, in which case an entirely different method of determination may be used. One possibility is for the application to use the HyTime BOS as a starting point, but to allow entities to be added and deleted by the user.

An SDIF data stream can carry any bounded object set, and is not strictly limited to hyperdocuments. In particular, the interchange data stream could include all of a single hyperdocument, a part of it, and all or part of other documents as well. In all cases, SDIF preserves the demarcation of documents and information objects, and can preserve the entity structure in which they existed in the originator's environment.

6.2.4.2 SDIF packer

The process (application or HyTime engine) that creates the BOS submits the names of the entities that contain the BOS to the SDIF packer program. The SGML document entity of the hub document, if present, is identified to SDIF as the "main" document. The packer creates an "entity descriptor" in the SDIF data structure for each entity. The descriptor includes the entity data. However, SDIF will create a "cross-reference" entity descriptor if the external identifier of an entity duplicates that of another, in order to avoid duplicating the data.

NOTE 72 The SDIF "packer" process that creates the data stream is aware of the entity declarations, and therefore of the media types of the entities. In a communications environment it would be possible to use this information to optimize bandwidth allocation and routing for different types of media when transmitting the hyperdocument.

HyTime provides a means of including the data of several entities within the storage of a single "container" entity, thereby allowing the use of interleaving and other techniques that optimize access to multimedia data. The data of the container entity is stored in its entity descriptor, while the entity descriptors of the contained objects are cross-references to the container (see 6.6).

6.2.4.3 SDIF unpacker

An SDIF unpacker receives an SDIF data stream and stores its contents in the storage media of the receiving system. Alternatively, it could act as a "handler" or "reader" service, invoked by the SGML entity manager, that unpacks SDIF dynamically as required by an application program.

If necessary, the SDIF unpacker modifies the system identifiers of the markup declarations to be consistent with storage addresses in its environment. When it encounters an entity declaration for which there is no entity descriptor (typically because it is outside the bounds of a HyTime BOS), it modifies the system identifier to show that the entity is not available. An attempt by an application to access that entity will result in an appropriate message.

NOTE 73 An SDIF unpacker implementation could choose to retain the sender's original external identifier, together with a "not received" indication and the date and time of the SDIF receipt. Later, if an unsuccessful attempt is made to access the entity, the entity manager could give the application the option of contacting the sender to request a copy, or trying to access it directly on the sender's system.

6.3 HyTime identification facilities

A document indicates conformance to HyTime by the APPINFO parameter of the SGML declaration, which specifies the name of the "HyTime prefix" that identifies SGML processing instructions that represent "HyTime declarations". HyTime declarations indicate the facilities required to support the document. Additional identification facilities are provided by architectural forms and attributes described in this sub-clause.

6.3.1 APPINFO parameter of SGML declaration

Conformance of a document to HyTime is indicated by a parameter of the APPINFO parameter of the SGML declaration. Its format is:

HyTime

The parameter can also specify the name of the "HyTime prefix" if it is other than "HyTime", and the name of the "HyTime attribute names" attribute (see 6.5.1) if it is other than "HyNames". The format is:

HyTime=HyTime,HyNames

where "HyTime" is replaced by the prefix name and "HyNames" is replaced by the name to be given to the "HyTime attribute names" attribute. Both names must be valid names in the concrete syntax of the SGML declaration.

6.3.2 HyTime support declarations

A HyTime document includes in its prolog one or more "HyTime support declarations" that collectively identify all of the facilities for which support may be required by the document. It is an error if a document requires a HyTime facility that is not identified in the support declarations, but a system is not required to report such an error as it may not be able to. However, it is not an RHE if the declarations identify facilities that are not actually required by the document.

If a system attempts to process a HyTime document whose support declarations identify facilities that the system does not support (see 12.6), the system must report an error even if the document does not actually use those facilities.

NOTE 74 In other words, a system can give the user the option of ignoring an "unsupported facility" error, but it must report the error in the first instance.

A HyTime support declaration is represented as an SGML processing instruction that begins with the HyTime prefix, followed by a "declaration name" and other parameters as required by the declaration type.

Each of the parameters must be preceded by one or more "white space" characters: that is, in SGML terms, RS, RE, SPACE, and SEPCHAR. White space characters can also follow the last parameter. The case of the parameters is not significant.

The HyTime declarations must appear in the document prolog, prior to the base document type declaration.

6.3.2.1 HyTime version declaration

The HyTime version declaration must be the first HyTime declaration in the document. It has the following format:

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32 >
```

where the number "32" represents a power of 2 that exceeds the highest quantum count for which support is required in the document. (See 7.4.2.3 for an explanation.)

6.3.2.2 HyTime module declarations

A HyTime module declaration is a HyTime declaration whose declaration name is "module". That name must be followed by one of the module names in the following list, thereby declaring that support is required for that module.

base	Base module
measure	Measurement module
locs	Location address module
links	Hyperlinks module
sched	Scheduling module
rend	Rendition module

If a module's optional facilities are employed, the module name must be followed by the names of the module's optional facilities for which support is being declared. If no options are specified, only support of the module's mandatory facilities is declared.

NOTE 75 For example, using the SGML reference concrete syntax, the following declarations declare that support is required for the non-optional facilities of the base module, the optional dvlist facility of the base module, and for the non-optional facilities of the location address module.

```
<?HyTime MODULE base dvlist >
<?HyTime MODULE locs >
```

Some options can be specified with an optional numeric value, as in:

```
<?HyTime MODULE sched manyaxes=3 >
```

If no value is specified, there is no limit to the number, as in:

```
<?HyTime MODULE sched manyaxes >
```

NOTE 76 If an option has mandatory components, such as mandatory attributes, they are required only if the option is supported.

If multiple support declarations name the same module, they are treated as a single combined declaration. Duplicate option names are ignored.

A summary of the names and descriptions of the options available in each module appears at the end of the corresponding clauses of this International Standard. In all cases, the description of a facility in the body of the clause will state any options on which the facility depends. All other facilities in the module are non-optional, and are not listed in the summary; support for them is indicated by support for the module.

If an option is dependent on another option in the same module, the dependency will be indicated in the option description in the summary, as follows:

```
HyLex    Allows HyLex models (requires: lextype).
```

Dependencies on other modules are shown in a similar fashion. It is an RHE to declare support for a facility that has dependencies without declaring support for the dependencies.

NOTE 77 A complete set of support declarations, with all options supported, is illustrated in 12.6.

6.4 HyTime document

The element type form **HyTime document** (*HyDoc*) represents the highest level element in the element structure of a HyTime document. Its content model, together with those of the other element type forms, comprise a meta-DTD to which all instances of conforming HyTime documents must conform. The attribute definitions in all of the HyTime attribute list forms are also part of the meta-DTD. It is a reportable HyTime error if a document instance fails to conform to the meta-DTD.

NOTE 78 The construct “+(%loc;%link;)” is called an “inclusion exception” in SGML. It means that the element types in parentheses can occur at any level of the element structure. In contrast, element types in the model group can occur only as “proper subelements”, in the immediate content.

The meta-DTD is flexible and does not clarify all issues of contextual relevance; some elements may be meaningful only in the context of other elements from which they are referenced, or only as resources for the processing of other elements. An application DTD can impose constraints on the meta-DTD by prohibiting the occurrence of some architectural forms in specific contexts.

NOTE 79 For example, it could allow elements conforming to hyperlink forms to occur only as proper subelements, only as inclusions, or not at all. As another example, an application DTD could define several element types, each conforming to a different architectural form that allows HyBrid content, but each having a different content model.

The parameter entity declarations are primarily an aid for tailoring the meta-DTD to correspond to the facilities supported by the HyTime system declaration. They also serve to document relationships among the architectural forms, but otherwise have no semantic significance.

NOTE 80 For example, if the "pathloc" option of the location address module is not supported, the meta-DTD can be modified easily by deleting the string "pathloc" (which corresponds to the option) from the entity declaration for "loc" (which corresponds to the facility).

The attribute **HyTime architectural form** (*HyTime*) identifies the architectural form of an element. Its value is the name of the architectural form to which the element conforms.

NOTE 81 A HyTime engine uses this attribute to determine the HyTime-specific processing to perform, and to know which HyTime-defined attributes to examine. The application processor supplements this processing by using the application-defined element type and other attributes to determine the application-specific processing to perform. In object-oriented programming terms, an application-defined subclass (the element type) inherits the properties of its object class (the architectural form).

The attribute **bounding object set level** (*bos/level*) specifies the bounding level of the HyTime BOS that is determined when the document is designated as a hub document. If not specified, the application determines its own BOS (see 6.2.4.1).

The attribute **unified name space** (*unmspace*) indicates whether entities and unique identifiers share a single name space ("unified") or not ("separate"). If unified is specified, it is an RHE if the unique identifier of an element is the same as an entity name in the DTDs or LPDs with respect to which the document is parsed, including entity names declared by means of a default entity declaration.

NOTE 82 This facility allows HyTime applications to define attributes whose values can be entity names, ID references, or both, without requiring support of the location address module.

The docmdu attributes are described in 7.4.2.3.

It is a reportable HyTime error if a document instance contains a value for an attribute defined in this or any other HyTime attribute list architectural form that fails to satisfy the declared value prescription defined for it in this International Standard.

NOTE 83 In particular, it is an RHE if a document instance contains a value for the HyTime architectural form attribute other than one specified in this International Standard. In other words, the set of HyTime architectural forms cannot be extended by an application.

```
<!-- BASE MODULE -->
    <!-- HyTime Document -->
<!ENTITY % loc      -- Location address architectural forms --
    "bibloc|nameloc|dataloc|treeloc|pathloc|listloc|relloc|proploc|notloc|fcsloc">
<!ENTITY % link     -- Hyperlink architectural forms --
    "click|ilink">
<!ENTITY % resbase  -- Resource architectural forms: base module --
    "activity|dvlist|desctab|lexmodel|lexord|propdef|qltn|qpn">
<!ENTITY % resmeas  -- Resource architectural forms: measurement module --
    "dimespec|extent|extlist|dimref|markfun|measure">
<!ENTITY % resschd  -- Resource architectural forms: scheduling module --
    "axis|exrecon|calspec|mallobj">
<!ENTITY % resrend  -- Resource architectural forms: rendition module --
    "rendrule|battrule|modrule|wandrule|wndpatch|modpatch">
<!ENTITY % resorce  -- Resource architectural forms: all modules --
    "(%resbase;)|(%resmeas;)|(%resschd;)|(%resrend;)">
<!ENTITY % HyBrid   -- HyBrid content (for use in content models) --
    "fcs|%resorce;|HyBrid|nHyTime|#PCDATA"
>
<!element HyDoc     -- HyTime document element --
    - O              (%HyBrid;)* +(%loc;|%link;) >
<!attlist HyDoc     HyTime      -- HyTime architectural form --
                                -- Unique value specified by standard
                                -- for each element type form --
                                NAME      HyDoc
                                id        ID          #IMPLIED -- Default: none --
                                boslevel -- Bounding level of HyTime bounded object set when
                                                document is a hub; overridable at run time --
                                                -- Constraint: 0=no limit; 1=root; 2 to N --
```

```

NUMBER    #IMPLIED    -- Default: application BOS --
unmspace  -- Unified name space for IDs and entities --
           (unified|separate) separate
           -- docmdu attributes --

```

>

6.4.1 HyTime/non-HyTime bridge element

The element type form **HyTime/non-HyTime bridge element** (*HyBrid*) bridges between HyTime and non-HyTime elements. No HyTime processing is defined for a HyBrid element (other than common attributes), but both HyTime and non-HyTime elements can occur in its content.

NOTE 84 HyTime elements in the content of a HyBrid element receive their normal HyTime processing.

```

<!-- HyTime/non-HyTime Bridge Element -->
<!element HyBrid    -- HyTime/non-HyTime bridge element --
                  -- Bridge between HyTime and non-HyTime elements --
                  - O      (%HyBrid;)* +(%loc;|%link;) >
<!attlist HyBrid    HyTime    NAME      HyBrid >

```

6.4.2 Non-HyTime element

The element type form **non-HyTime element** (*nHyTime*) is one for which no HyTime processing occurs at any level of its content, except for the recognition and processing of common attributes.

NOTE 85 The suppression of HyTime processing can be useful if HyTime elements are to be used for non-HyTime purposes; for example, to display or edit HyTime resource elements.

The attribute definition list for a non-HyTime element type that can only occur within the content of another non-HyTime element does not require the "HyTime" attribute unless one or more HyTime common attributes are also defined. If the "HyTime" attribute is not defined, a HyTime engine will not recognize common attributes for that element type.

NOTE 86 In other words, the HyTime attribute needs to be defined for an nHyTime element type only if:

- the element type could establish a "no HyTime processing" mode (that is, the element type could occur other than within nHyTime elements that would already have established that mode); or
- the element type has HyTime common attributes that must be recognized and processed by the HyTime engine.

NOTE 87 Note that, unlike a HyBrid element type, an nHyTime element type can contain any HyTime elements, even those that normally can occur only in the content of some other HyTime element. This freedom is possible because the HyTime processing is suppressed, and therefore does not require the processing context that would have been established by the containing element.

```

<!-- Non-HyTime Element -->
<!element nHyTime  -- Non-HyTime element --
                  -- Establishes "no HyTime processing" mode for all levels
                  -- of its content, except for common attributes --
                  - O      ANY >
<!attlist nHyTime  HyTime    -- Defined for nested nHyTime elements only if
                           common attributes are defined for element type --
                           NAME      nHyTime

```

>

6.5 Common attributes

6.5.1 Identification attributes

The attribute list form *all-id* consists of common attributes that identify an element, its content, and its attributes. These common attributes can be used individually in attribute list definitions; each is considered a separate facility from the standpoint of determining whether it is mandatory or non-mandatory.

The attribute **unique identifier** (*id*) uniquely identifies an element in the name space of the document. This is the normal SGML usage; there are no special HyTime semantics. As the attribute has a declared value prescription of "ID", it will be recognized even if no "HyTime" attribute is defined for the element.

NOTE 88 SGML subdocuments have their own ID name spaces.

The attribute **data content notation** (*notation*) identifies the notation in which the data content of the element is represented. The attribute is processed in the same way that an SGML parser treats a notation attribute. If the attribute is defined with a declared value prescription of "NOTATION" (rather than "NAME"), it will be recognized even if no "HyTime" attribute is defined for the element.

A data content notation could be defined with delimiter roles whose string assignments can be changed for individual element types or elements. The attribute **notation delimiter set** (*delims*) specifies, for one or more notation delimiter role names, the new character strings to be assigned.

NOTE 89 This attribute does not change the SGML delimiter set.

The attribute **content location** (*conloc*) identifies the content of the element when it is not in the syntactic content (that is, between the start- and end-tags).

NOTE 90 The keyword "#CONREF" in the SGML attribute definition identifies this attribute as a "content reference attribute". If a value is specified for a content reference attribute, there must be no content within the element. If no value is specified, content will be expected in the usual place (after the start-tag).

If the located content is data or an element that satisfies the content model of this element, it will be used as the content of this element. If not, and the located content is an element whose own content satisfies the content model of this element, then the content of the located element will be used as the content of this element. It is an RHE if neither of the above conditions is met.

NOTE 91 For example, if the located element were an ordered list whose content is a sequence of list item elements, the ordered list element would be treated as the content of this element if it satisfied the content model. If it did not, the list items would be treated as the content if they satisfied this element's content model.

The attribute **contextual content** (*context*) indicates whether the content of the element (if any) must be valid in the context in which the element occurs, in both the DTD and the meta-DTD.

NOTE 92 In other words, the content will be parsed as though the element itself did not exist.

The context attribute requires support of the "context" option.

The attribute **HyTime attribute names** (*HyNames*) defines user substitutes for HyTime attribute names. The substitute is treated as if it were the correspondingly-named HyTime attribute. If the substitute name is "#CONTENT", the value of the HyTime attribute is the syntactic content of the element. An attribute bearing the HyTime attribute name for which a substitute is defined will be ignored by HyTime.

NOTE 93 The attribute name "HyNames" can itself be renamed by the APPINFO parameter of the SGML declaration (see 6.3.1).

```
<!-- Identification -->
<!attlist all-id
    id          -- SGML unique identifier --
               ID          #IMPLIED -- Default: none --
    notation    -- Data content notation --
               NAME        #IMPLIED -- Default: none --
    delims      -- Data content notation delimiter changes --
```

```

-- Constraint: names are delimiter role names --
-- Constraint: a given NAME can occur only once --
-- lextype( (NAME, s+, char*),
            (s+, NAME, s+, char*)* ) --
CDATA      #IMPLIED -- Default: not changed --
conloc      -- Content location (if not in syntactic content) --
-- Constraint: data or element that satisfies
            this element's content model, or element whose
            content satisfies this element's content model.--
IDREFS      #CONREF -- Default: syntactic content --
context     -- Contextual content --
            (context|ncontext) ncontext
HyNames     -- User names for HyTime attributes --
-- Constraint: a given ATTNAME or #CONTENT
            can occur only once --
-- lextype(ATTNAME, s+, (NAME|(RNI,"CONTENT")),
            (s+, ATTNAME, s+, (NAME|(RNI,"CONTENT")))* ) --
CDATA      #FIXED "" -- Default: none --

```

>

6.5.2 Lexical types

A "lexical type" is a defined pattern of characters or tokens to which a character string or token sequence, respectively, conforms. A definition of a lexical type is called a "lexical model". Data that satisfies a test for conformance to a lexical model is called a "hit".

The attribute list form *all-lex* consists of common attributes that allow an application to require that attributes and/or content must conform to a lexical type. The form requires support of the "lextype" option. Both of its attributes must be defined if the form is used.

The attribute **lexical types** (*lextype*) identifies those attributes of an element, and/or its content, that must conform to a named lexical model. The attributes and/or content are declared as some form of character data. Normal SGML parsing of the character data occurs prior to lexical parsing.

The attribute **lexical model** (*lexmodel*) specifies the lexical model to which each attribute specified in *lextype* conforms.

NOTE 94 The *lextype* and *ulextype* conventional comments perform the function of *lextype* and *lexmodel* for some attribute value and content strings defined by the HyTime language. Care should be taken when defining *lextype* and *lexmodel* attributes that constrain such strings to assure that the constraints of the *lextype* and *ulextype* comments can still be satisfied.

In both *lextype* attributes and *lextype* conventional comments, a value of the property "ATTNAME" must be an attribute name in the attribute definition list to which the *lextype* applies.

HyTime defines a property (see 6.7) for attributes and data content called the "lexical type" (*lextype*). Its value is the lexical type defined for the attribute or content by a *lextype* conventional comment or a *lextype* attribute. If neither is specified, the value is the declared value prescription for the attribute. For the content, it is the declared content (CDATA or RCDATA), if any, or "PCDATA".

```

<!-- Lexical Types -->
<attlist all-lex
    lextype -- Lexical types of attribute values
            or character data content --
            -- Constraint: a given ATTNAME or #CONTENT
            can occur only once; a qltn must be an ltn of a
            lexmodel whose GI is specified in lextypgi --
            -- lextype(ATTORCON, s+, QLTN,
            (s+, ATTORCON, s+, QLTN)* ) --
            CDATA      #FIXED "" -- Default: none defined --
    lextypgi -- Lexical model element types used by lextype --
            -- Constraint: searched in order listed; must
            be GIs of lexmodel form element types --

```

```

-- lextype(GIL) --
NAMES      #FIXED in-DTD
>

```

6.5.2.1 Lexical model

The element type form **lexical model** (*lexmodel*) represents a lexical model.

NOTE 95 A lexical model element type and corresponding data content notation, both known as "HyLex", are specified in A.1. They require support of the "HyLex" option.

The attribute **lexical type name** (*ltn*) identifies an individual model for use in a *lextype* attribute, or as a component of another model, for those notations that allow it. An *ltn* must be unique among all lexical model elements of the same type.

As a notation could allow intermixing of *ltn* references from several notations, the specification of an *ltn* can be qualified by the lexical model GI in order to avoid ambiguity if identical lexical type names of two different element types are used in the same model. For this reason, the specification of an *ltn* is known as a "qualified lexical type name" (*qltn*). It is normally represented as a character string conforming to the lexical type "*qltn*", as defined in A.1.3.

The attribute **lexicographic ordering definition** (*lexord*) specifies the lexicographic ordering to be used when testing data for conformance to the lexical model (see 6.5.2.2). The reserved name "#SGMLCASE" identifies an ordering in which all name characters (not just those being tested against properties based on SGML names) are treated as their upper-case counterparts, as defined in the SGML declaration. The *lexord* attribute requires support of the "lexord" option.

The attribute **hit boundary constraint** (*boundary*) specifies the boundary constraints, within the match domain, of a hit that satisfies the model, as follows:

- sodeod The hit must extend from the start of domain to the end.
- sodiec The hit must begin at the start of domain but need not extend to the end of domain.
- isceod The hit must end at the end of domain but need not begin at the start of domain.
- isciec The hit can occur anywhere in the match domain.
- inmodel Boundary requirements are expressed within the model.

When a lexical model is used for testing the existence of hits, as in a query language, the usual boundary constraint is "isciec". When a model is used to prescribe the lexical type of data, as with the *lextype* attribute, the usual boundary constraint is "sodeod". In the conventional comments that define lexical types for the attributes and data content in this International Standard, "sodeod" is the boundary constraint.

NOTE 96 If the content of a lexical model is data, a data content notation attribute would normally be fixed in the DTD to allow the SGML parser to determine the program that interprets the lexical model. Alternatively, the generic identifier of the element type could provide sufficient information for the application to locate the interpreter itself. The application is also responsible for interpretation when the content consists of subelements (although the subelements could in turn have content that is subject to a data content notation).

```

<!-- Lexical Model -->
<!element lexmodel -- Lexical model --
  - O      (%HyBrid;)* >
<!attlist lexmodel HyTime      NAME      lexmodel
  ltn      -- Lexical type name --
           -- Constraint: unique in element type --
           NAME      #REQUIRED
  lexord   -- Lexicographic ordering for comparisons --
           -- reftype(lexord) --
           -- lextype(IDREF| (RNI,"SGMLCASE")) --
           CDATA     #IMPLIED -- Default: character set order--
  boundary -- Hit boundary constraint --

```

```
(sodeod|sodiec|isceod|isciec|inmodel) sodeod
```

>

6.5.2.2 Lexicographic ordering definition

Characters in a computer system are inherently ordered by their coded representations, which assign them positions in a character set.

NOTE 97 The base-10 integer representation of a coded representation is called a "character number". In an SGML document, characters can be represented by their coded representations (bit combinations) in the usual way, and also by means of a string containing a delimited character number, called a "numeric character reference".

When characters are compared, as when sorting, testing conformance to a lexical model, or testing equality in a query, the character set order may not be suitable. The element type form **lexicographic ordering definition** (*lexord*) allows alternative positions to be assigned for such comparisons.

NOTE 98 The alternative positions are sometimes called a "sort order", although their use is not restricted to sorting. The lexicographic ordering definition is sometimes called a "sort translation", although in fact no permanent character number translation occurs: the reordering is normally transitory and internal to the processing program.

The *lexord* facility is not restricted to one-to-one reorderings. The element type form **grapheme** (*grapheme*) associates a string of one or more characters, identified by the attribute **grapheme string** (*gs*), with one or more reordered character numbers, identified by the attribute **grapheme comparison key** (*gck*). If several grapheme strings begin with the same character sequence, the longest string is recognized.

NOTE 99 The following example of a fragment of a *lexord* element illustrates one-to-many and many-to-one reorderings.

```
<grapheme gs= a      gck= 1      >
<grapheme gs= d      gck= 4      >
<grapheme gs= h      gck= 8      >
<grapheme gs= j      gck= " 4 27 8" ><!-- j compares as dzh -->
<grapheme gs= l      gck= 12     >
<grapheme gs= ll     gck= 13     ><!-- ll occurs between l and m -->
<grapheme gs= m      gck= 14     >
<grapheme gs= &apos;   gck=       ><!-- apostrophe ignored in compare -->
<grapheme gs= z      gck= 27     >
```

```
<!-- Lexicographic Ordering Definition -->
<!-- element lexord -- Lexicographic Ordering Definition --
- O      (grapheme+) >
<!-- attlist lexord
HyTime  NAME      lexord
id      ID        #REQUIRED
>
<!-- element grapheme -- Grapheme comparison key --
- O      (%HyBrid;)* >
<!-- attlist grapheme
HyTime  NAME      grapheme
gs      -- Grapheme string (case-sensitive) --
        -- Constraint: must be unique in lexord --
        CDATA      #REQUIRED
gck     -- Grapheme comparison key --
        NUMBERS    #IMPLIED -- Default: ignored in compare --
>
```

6.5.3 ID reference resolution control

The attribute list form *all-ref* allows an application to control the target and resolution range of ID references. The form requires support of the "refctl" option.

An ID reference is an attribute whose declared value prescription is IDREF or IDREFS, or an attribute or data content that is declared to contain ID references by a lextype attribute whose *lexmodel* is HyLex. Attribute ID

references are specified by the attribute names, and content ID references by the reserved name "#CONTENT". The reserved name "#ALL" means "all ID references in the attribute values and data content of this element type".

The attribute **ID reference resolution range** (*refrange*) associates the names of ID references with resolution ranges. The resolution range associated with "#ALL" can be overridden for individual ID references by specifying their names with different ranges.

There are four resolution ranges, listed in order of increasing freedom; that is, the later ranges in the list also permit resolutions allowed by the ranges that precede them:

- B **Backward direct reference:** An identified local element that precedes the reference in the same document (or subdocument).
- D **Direct reference:** An identified local element in the same document (or subdocument) as the reference.

NOTE 100 A direct reference can be either backward or forward.

- I **Indirect reference:** A location in the same document (or subdocument) as the reference that is referenced indirectly through a location address element. This resolution range requires support of the location address module. If that module is not supported, this resolution range is treated like the "D" range, rather than being treated as an RHE.
- X **External reference:** A location in any document (or subdocument). This resolution range requires support of the "exidrefs" option. If that option is not supported, this resolution range is treated like the "I" range, rather than being treated as an RHE.

NOTE 101 When the hyperlinks module is used, the *refrange* attribute allows all references to be represented as HyTime links without requiring purely local ones (such as footnote references) to incur the performance penalties of potentially universal reference. To put it another way, it allows purely local links to be optimized.

The attribute **ID reference resolution level** (*reflevel*) associates the names of ID references with maximum resolution levels of external or indirect references. The attribute requires support of the location address module.

The resolution level associated with "#ALL" can be overridden for individual ID references by specifying their names with different levels. An ID reference for which no maximum resolution level is specified (either by name or via "#ALL") is fully resolved.

A resolution level is specified in the form of an unsigned non-zero integer that represents the maximum allowable level of ID reference resolution relative to this element. If the target of a reference is itself a reference with a *reflevel* specified (or its target is, and so on), resolution is terminated at the earliest point specified by any of them.

NOTE 102 Some examples:

- If an element A has an IDREF with a *reflevel* of 6, resolution will cease 6 levels below it (7th level overall).
- If element B is a target of A's reference and has an IDREF with a *reflevel* of 3, resolution will cease 3 levels below B (4 levels below A, 5th level overall).
- If element C is a target of B's reference and has an IDREF for which no *reflevel* is specified (unlimited), resolution will cease 2 levels below C (4 levels below A, 5th level overall) because the earliest termination point specified (that of element B's IDREF) governs.

NOTE 103 The indirect ID reference pointers, from the initial reference to the target locations, comprise a "location ladder" that is normally transparent to the applications that process the target objects. The transparency of the indirect referencing allows the location ladder to be revised when an object is moved, without affecting the application. When a *reflevel* is specified, however, the rungs of the location ladder are addressed, and this transparency may be lost.

The attribute **ID reference element type** (*reftype*) associates names of ID reference attributes (or #CONTENT) with element types to which their targets must conform. The element types can be specified as GIs, or the reserved word "#ANY", which signifies that any target is acceptable for the corresponding attribute. Alternatively, a number can be specified, which signifies that a specific number of targets are required, but that they need not conform to the content model of an element type. The constraints associated with "#ALL" can be overridden for particular ID reference attributes by specifying element types or numbers for them individually.

The constraints are those of the target of the reference; intermediate location addresses can be used if permitted by the `refrange` attribute. If the constraint is an element type, the target must be either an element with the specified GI, or data and/or elements that satisfy the content model of the element type with the specified GI.

NOTE 104 This provision allows an application, for example, to specify that the target is one of a set of alternatives: the reftype value names the GI of an element (defined for the purpose) that has an OR group for its content model. The ID reference identifies an element that is a member of the OR group.

Normally, the targets of a list of ID references must each satisfy the reftype. However, if `"#SEQ"` is specified, they are treated as a sequence to do so. That is, the targets of the ID reference list, taken in the order that the list is specified, must collectively satisfy the content model of the specified GI.

NOTE 105 For example, if the reftype specification is `"#ALL A #SEQ"` and A's content model is `"(B,C)"`, the reftype is satisfied by a list of two ID references, the first of whose target is `"B"` and the second `"C"`. The reftype is also satisfied by a single ID reference whose target is `"B"` followed by `"C"`.

NOTE 106 The keyword `"#SEQ"` should not be specified if the content model of the specified GI has only a single token.

If `"#SEQ"` is specified when the constraint is a number (rather than a GI), each IDREF must correspond to exactly one target (that is, any intermediate location addresses cannot be multiple or empty). This requirement does not prevent a target from being an aggregate anchor of a hyperlink, as an aggregate anchor is an object in this context, not an intermediate location address.

NOTE 107 If `"#SEQ"` is specified when indirect references are not permitted, the lextyp or declared value prescription must allow a sufficient number of IDREFS. However, if indirect references *are* permitted, no more than one of the IDREFS can be required, as one multiple location address element could locate all the targets needed to satisfy the model.

NOTE 108 The reftype attribute allows attribute values to be structures. The structures are defined as SGML element types, and instances of the structures are represented as elements. The structures do not occur directly in the attribute values; instead, ID references are used to point to them. The application designer can use the `refrange` attribute to require that the structures are referenced without indirect location addresses, and/or that they precede the reference in the document, in order to keep them close to the start-tags and to allow efficient processing.

NOTE 109 The reftype conventional comment performs the function of a "meta" reftype (with somewhat different syntax) for ID references defined by the HyTime language, as it constrains the architectural forms of ID reference targets, not their element types. An application designer can impose further constraints by specifying reftype attributes that will restrict the target element types.

HyTime defines a property (see 6.7) for ID reference attributes and content called the "reference type" (reftype). Its value is the reftype defined for the attribute or content by a reftype conventional comment or a reftype attribute.

Failure of ID references to conform to the ID reference resolution control attributes (`refrange`, `reftype`, etc.), or to conform in number or type to other attributes, are RHEs only when (and if) the application requires the IDREFS to be resolved sufficiently to recognize the failure. Such failures are always RHEs, however, if they can be determined from the markup alone (for example, if the number of IDREFS specified differs from the number prescribed by a reftype in the form `"number #SEQ"`).

```
<!-- ID Reference Resolution Control -->
<attlist all-ref
    refrange -- ID reference resolution range --
        -- B Backward reference to identified local element
        D Identified local element, before or after
        I Location address allowed
        X Indirect target, external to reference --
    -- Attribute declared value or lextyp is IDREF --
    -- Constraint: a given ATTNAME or #CONTENT
    can occur only once --
    -- lextyp( (RNI,"ALL", s+, ("B"|"D"|"I"|"X"))?,
        (s+, ATTORCON,
            s+, ("B"|"D"|"I"|"X"))* ) --
    CDATA #FIXED "#ALL I"
    refllevel -- ID reference resolution level --
    -- Level that IDREF resolution cannot exceed,
    relative to this element. --
```