

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

ICS 35.060; 35.240.99

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

National foreword

This British Standard reproduces verbatim ISO/IEC 10744:1997 and implements it as the UK national standard. It supersedes BS ISO/IEC 10744:1992 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/18, Document Processing and Related Communication, which has the responsibility to:

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

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international or European publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

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

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, the ISO/IEC title page, pages ii to x, pages 1 to 373 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

Amendments issued since publication

Amd. No.	Date	Comments

This British Standard, having been prepared under the direction of the DISC Board, was published under the authority of the Standards Board and comes into effect on 15 May 1998

© BSI 04-2000

ISBN 0 580 29802 7

Contents

	Page
National foreword	Inside front cover
Foreword	x
Text of ISO/IEC 10744	1

INTERNATIONAL STANDARD

**ISO/IEC
10744**

Second edition
1997-08-01

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

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



Contents

	Page
Foreword	x
Introduction	1
1 Scope	4
1.1 Definition of scope	4
1.2 Field of application	4
2 Normative references	5
3 Definitions	5
4 Symbols and Abbreviations	17
5 Notation	17
5.1 RCS name, full name, description, and clause	18
5.2 Lexical type	18
5.3 Constraints	18
5.4 Note	18
5.5 Associated attribute forms and attribute lists	18
5.6 Referrers	19
5.7 Conventions for attribute form declarations	19
5.8 Identification of optional facilities	19
6 Base module	19
6.1 Concepts and definitions	19
6.1.1 Object representation	19
6.1.1.1 Entity structure	20
6.1.1.2 Data	21
6.1.2 Object identification and addressing	21
6.1.2.1 Name space addressing	21
6.1.2.2 Coordinate addressing	21
6.1.2.3 Semantic addressing	22
6.2 Hyperdocument management facilities	22
6.2.1 Object representation	22
6.2.2 Object identification and addressing	22
6.2.3 Object access	23
6.2.4 Bounded object set (BOS)	23
6.2.5 Hyperdocument interchange format	25
6.2.5.1 SDIF packer	25
6.2.5.2 SDIF unpacker	25
6.3 HyTime support declarations	26
6.4 HyTime document	28
6.5 HyTime Bounded object set	29
6.5.1 HyTime bounded object set attributes	29
6.5.2 HyTime BOS control data attributes	30
6.5.3 Bounded object set exception specification	32
6.6 HyTime architectural bridging forms	34
6.7 Common attributes	35
6.7.1 Value Reference	35
6.7.2 Descriptive text	37
6.7.2.1 Descriptive text attributes	37
6.7.2.2 Description table	37
6.7.2.3 Descriptive text	38
6.7.2.4 Descriptive text definition	38
6.7.3 Activity policy association	39

		Page
6.8	Coordinate Specifications	44
6.8.1	HyTime axis marker list notation	45
6.8.1.1	Axis marker list	45
6.8.1.2	Marker Functions	45
6.8.2	HyTime dimension specification notation	46
6.8.3	Dimension Specification	47
6.8.4	Dimension List	48
6.8.5	Overrun handling	48
6.8.6	HyTime Marker Function Language (HyFunk)	49
7	Location address module	51
7.1	Concepts and definitions	51
7.1.1	Location types	51
7.1.2	Location Sources	52
7.1.3	Location Paths	53
7.1.4	Groves and Location Addressing	54
7.1.4.1	Grove Plan	54
7.1.4.2	HyTime Default SGML Grove Plan	57
7.1.4.3	Effective SGML Document Grove Plan	58
7.1.4.4	Grove Definition Elements	59
7.2	Location source	60
7.3	Implied location source	61
7.4	Multiple location	62
7.5	Tree type	62
7.6	Span Location Address	63
7.7	Reference control	64
7.7.1	Reference element type	64
7.7.2	Reference resolution range	65
7.7.3	Reference resolution level	65
7.8	Reference location address	66
7.9	Name-space locations	69
7.9.1	Identified local element or entity	69
7.9.2	Property location address	69
7.9.3	Name-space location address	71
7.9.4	Mixed location address	72
7.9.5	Named location address	72
7.9.6	Name list specification	73
7.10	Coordinate locations	74
7.10.1	Node locations	74
7.10.1.1	Node lists	74
7.10.1.2	List location address	74
7.10.1.3	Tree combination	75
7.10.1.4	Tree location address	75
7.10.1.5	Path location address	76
7.10.1.6	Relative location address	77
7.10.2	Data location address	79
7.11	Querying	84
7.11.1	Query location address	84
7.11.2	Name list query	85

	Page
7.12	Bibliographic location address
8	Hyperlinks module
8.1	Concepts and definitions
8.1.1	Link creation
8.1.2	Link traversal
8.1.3	Traversal Rules
8.2	Hyperlink architectural forms
8.2.1	Hyperlink
8.2.2	Contextual link
8.2.3	Aggregation link
8.2.4	Variable link
8.2.5	Independent link
8.3	Hyperlink-related location addresses
8.3.1	Hyperlink location address
8.3.2	Hyperlink anchor location address
9	Scheduling module
9.1	Scheduling concepts and definitions
9.2	Measurement units
9.2.1	Measurement domain definition
9.2.2	HyTime granule definition notation
9.2.3	Useful measurement domains
9.2.3.1	Common Standard Measurement Units
9.2.3.2	Measurement domain definitions
9.2.3.3	Other standard measurement units
9.3	Finite coordinate space
9.3.1	Axis calibration
9.4	Scheduling and extents
9.4.1	Schedules
9.4.2	Extent specification
9.4.3	Group extent specification
9.4.4	Scheduled extent
9.4.5	Scheduled extent list
9.4.6	HyTime extent list notation
9.5	Event schedule
9.5.1	Scheduled Event
9.5.2	Event group
9.6	Objects
9.7	Pulse maps
9.8	Dimension referencing
9.8.1	Implicit dimension reference
9.8.2	Explicit dimension reference
9.8.2.1	Referencing dimensions of directly scheduled events, modscopes, and/or proscopes
9.8.2.2	Referencing dimensions of indirectly scheduled events, modscopes, and/or proscopes
9.9	Calibrated real time axes
9.9.1	HyTime calendar specification notation
9.9.2	Calendar specification

		Page
9.10	Finite coordinate space location address	132
10	Rendition module	135
10.1	Common rendition attributes	135
10.1.1	Precision of Selection	135
10.2	Object Modification	137
10.2.1	Object modifier	137
10.2.2	Direct association of modifiers (modifier rule)	137
10.2.3	Association of modifiers by position in finite coordinate spaces	138
10.2.3.1	Wand Rule	138
10.2.3.2	Wand	139
10.2.3.3	Modifier scope	139
10.2.3.4	Modifier scope group	140
10.2.4	Modifier Patch and Wand Patch	141
10.3	Projection	155
10.3.1	Projector	143
10.3.1.1	HyTime Projector Notation	144
10.3.1.2	Extent Projector	145
10.3.1.3	HyTime Extent Projector Notation	145
10.3.1.4	Dimension Projector	146
10.3.1.5	HyTime Dimension Projector Notation	147
10.3.2	Direct association of projectors	148
10.3.2.1	Projection of modified and unmodified objects	148
10.3.2.2	Projector Rule	148
10.3.2.3	Projector sequence	149
10.3.3	Association of projectors by position in finite coordinate spaces	150
10.3.3.1	Baton rule	150
10.3.3.2	Baton	151
10.3.3.3	Projector scope	151
10.3.3.4	Projector scope group	152
10.3.3.5	Baton sequence	152
10.4	Rendition rule	153
11	Conformance	153
11.1	Conforming HyTime document	153
11.1.1	Basic hyperlinking HyTime document	153
11.1.2	Basic scheduling HyTime document	154
11.1.3	Minimal HyTime document	154
11.1.4	Minimal scheduling HyTime document	155
11.2	Conforming HyTime application	155
11.2.1	Application conventions	155
11.2.2	Conformance of documents	155
11.2.3	Conformance of documentation	155
11.3	Conforming HyTime system	155
11.3.1	Conformance of documentation	156
11.3.2	Conformance to HyTime system declaration	156
11.3.3	Support for minimal HyTime documents	156
11.3.4	Application conventions	156
11.4	Validating HyTime engine	156
11.4.1	Error recognition	156

		Page
11.4.2	Identification of HyTime messages	156
11.4.3	Content of HyTime messages	156
11.5	Documentation requirements	156
11.5.1	Standard identification	157
11.5.2	Identification of HyTime constructs	157
11.5.3	Terminology	157
11.6	HyTime system declaration	157
Annex A (normative) SGML Extended Facilities		159
A.1	Introduction	159
A.1.1	Conformance	159
A.1.1.1	Application conventions	159
A.1.1.2	Conformance of documents	159
A.1.1.3	Conformance of documentation	159
A.1.1.3.1	Standard identification	159
A.1.1.3.2	Identification of Extended Facilities constructs	160
A.1.1.3.3	Terminology	160
A.1.1.3.4	Application conventions	160
A.2	Lexical Type Definition Requirements (LTDR)	160
A.2.1	Lexical type set	160
A.2.1.1	Lexical types	161
A.2.1.2	Lexicographic ordering	162
A.2.1.3	Additional lexical constraints	163
A.2.2	Lexical model notations	163
A.2.3	HyTime lexical model notation (HyLex)	164
A.2.3.1	Syntax	164
A.2.3.2	Normalized HyLex models	165
A.2.3.3	Intrinsic lexical types	165
A.2.4	Lexicographic ordering definitions notations	177
A.2.4.1	HyTime lexicographic ordering definition notation (HyOrd)	177
A.3	Architectural Form Definition Requirements (AFDR)	178
A.3.1	Enabling architectures	178
A.3.1.1	Architectural forms	179
A.3.1.2	Architectural document	180
A.3.2	SGML conventions	180
A.3.2.1	Element forms	180
A.3.2.1.1	Element type declaration	180
A.3.2.1.2	Meta-DTD	181
A.3.2.1.3	Attribute definition list declaration	182
A.3.2.2	Attribute forms	182
A.3.2.3	Attribute list conventions	183
A.3.2.3.1	Default value prescription	183
A.3.2.4	Processing link attributes	184
A.3.3	Architecture base declaration	184
A.3.3.1	Enabling architecture use of APPINFO parameter	184
A.3.4	Architecture support declarations	185
A.3.4.1	Architecture notation declaration	185
A.3.4.2	Architecture support attributes	185
A.3.4.3	Architecture entity declaration	188

		Page
A.3.5	Architecture control attributes	189
A.3.5.1	Architectural form attribute	189
A.3.5.2	Architectural attribute renamer	189
A.3.5.3	Architecture suppressor attribute	191
A.3.5.4	Architecture ignore data attribute	191
A.3.6	Other architecture-related considerations	192
A.3.6.1	Architectural document element	192
A.3.6.2	Architectural markup minimization	192
A.3.6.3	Derived enabling architectures	193
A.3.6.4	Relating applications and architectures	193
A.3.7	Summary of AFDR support options	193
A.3.8	Conformance	193
A.3.8.1	Conformance of meta-DTDs	193
A.3.8.2	Conformance of documents and derived meta-DTDs	194
A.3.8.3	Conforming architecture engine	194
A.3.8.3.1	Conformance of documents	194
A.3.8.3.2	Conformance of documentation	194
A.3.8.3.3	Application conventions	194
A.3.8.4	Validating architecture engine	194
A.3.8.4.1	Identification of architecture messages	194
A.3.8.4.2	Content of architecture messages	194
A.3.8.5	Architecture system declaration	194
A.4	Property Set Definition Requirements (PSDR)	195
A.4.1	Concepts and terminology	195
A.4.1.1	Property sets	195
A.4.1.2	Classes and properties	196
A.4.1.3	Nodes	197
A.4.1.4	Groves	197
A.4.1.5	Content trees	197
A.4.1.6	Grove plan application	198
A.4.2	Property set definition architecture	198
A.4.2.1	Shared constructs	200
A.4.2.1.1	Component names	200
A.4.2.1.2	Specification and clause	201
A.4.2.1.3	Descriptive elements	201
A.4.2.1.4	Member of default grove plan	201
A.4.2.2	Modules	202
A.4.2.3	Class definition	202
A.4.2.4	Property definition	203
A.4.2.4.1	Enumerated value definition	205
A.4.2.5	Normalization rule definition	205
A.4.3	Intrinsic properties	205
A.4.4	Useful grove construction processes	207
A.4.4.1	Value-To-Node (VTN) grove construction	207
A.4.4.1.1	The Value-To-Node property set	207
A.4.4.1.2	VTN groves	208
A.4.4.2	Data tokenizer (DATATOK) grove construction	209

	Page
A.4.4.2.1 Data tokenizer property set	210
A.4.4.2.2 Data tokenizer notation form	210
A.4.4.3 Plain text (PLAINTEXT) grove construction	211
A.4.4.3.1 Plain text property set	212
A.4.5 Canonical Grove Representation (CGR)	212
A.4.5.1 Canonical grove representation document Type	212
A.4.5.2 Constraints on CGR source construction	214
A.4.5.3 Algorithm for assigning IDs to nodes	215
A.4.6 Conformance	216
A.5 General Architecture	216
A.5.1 General Architecture Declaration Template	216
A.5.2 Common attributes of elements	217
A.5.3 Data Attributes for Elements (DAFE)	218
A.5.3.1 Data control attributes	219
A.5.4 Lexical types	219
A.5.5 ID immediate referent type control	220
A.5.6 Default value list	221
A.5.6.1 Default value list attributes	221
A.5.6.2 Default value list element	222
A.5.7 Data attributes	223
A.5.7.1 Common data attributes	223
A.5.8 Conformance	224
A.5.8.1 Conforming General Architecture document	224
A.5.8.1.1 Minimal General Architecture document	224
A.5.8.2 Conforming General Architecture application	224
A.5.8.2.1 Application conventions	224
A.5.8.2.2 Conformance of documents	225
A.5.8.2.3 Conformance of documentation	225
A.5.8.3 Conforming General Architecture system	225
A.5.8.3.1 Conformance of documentation	225
A.5.8.3.2 Conformance to General Architecture system declaration	225
A.5.8.3.3 Support for minimum General Architecture documents	225
A.5.8.3.4 Application conventions	225
A.5.8.4 Validating General Architecture engine	225
A.5.8.4.1 Error recognition	225
A.5.8.4.2 Identification of General Architecture messages	225
A.5.8.4.3 Content of General Architecture messages	226
A.5.8.5 Standard identification	226
A.5.9 General Architecture system declaration	226
A.6 Formal System Identifier Definition Requirements (FSIDR)	227
A.6.1 System identifiers	227
A.6.1.1 Storage object specification (SOS)	227
A.6.1.2 Informal system identifiers	227
A.6.1.3 Entity usage attributes	228
A.6.2 Auxiliary processes	228
A.6.3 Containers	229
A.6.4 FSI identification facilities	229

	Page
A.6.4.1 FSI use of APPINFO parameter	229
A.6.4.2 FSI declaration	230
A.6.4.3 FSI syntax	230
A.6.5 Storage manager attribute definitions	231
A.6.5.1 Record-related attributes	231
A.6.5.2 Encoding-related attributes	232
A.6.5.2.1 Encoding notations	233
A.6.5.2.2 BCTF algorithm notations	235
A.6.5.3 Common storage manager attributes	236
A.6.6 Entity usage attribute definitions	237
A.6.7 Storage manager notation forms	239
A.6.7.1 Local storage manager notation form	239
A.6.7.2 Starter set local storage managers	239
A.6.7.3 Portable storage manager notation form	240
A.6.7.3.1 URL Portable storage manager	241
A.6.7.3.2 Neutral file identifier storage manager	241
A.6.7.3.3 Notation processor storage manager notation form	242
A.6.7.3.4 Notation processor storage managers	242
A.6.7.4 Global storage manager notation form	243
A.6.7.5 Global storage managers	243
A.6.7.6 Container storage manager notation form	244
A.6.7.6.1 Container storage managers	245
A.6.7.6.2 Standard BENTO (sbento)	246
A.6.8 Conformance	247
A.7 SGML Property Set	248
A.7.1 SGML Notation	248
A.7.2 SGML property set	249
Annex B (normative) HyTime Property Set	280
B.1 Hyperdocuments and HyTime documents	280
B.2 HyTime Property Set	280
Annex C (normative) Architectural Meta-Declarations	294
C.1 HyTime Lexical Types	294
C.1.1 Calendar-Related Lexical Types	296
C.2 HyTime Meta-Declarations	297
C.3 General Architecture Meta-Declarations	367
Annex D (informative) Supplementary materials	373

Descriptions: Data processing, text processing, information interchange, artificial languages, programming languages, Hytime.

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 JTC1, *Information technology*, Subcommittee SC 18, *Document processing and related communication*.

This second edition cancels and replaces the first edition (ISO/IEC 10744:1992), which has been technically revised.

Annex A, Annex B and Annex C form an integral part of this International Standard. Annex D 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, in a variety of ways. 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 are expressed as an “enabling architecture”, consisting of a number of “architectural forms” and their associated semantics. The HyTime standard’s formal definition as an architecture conforms to the Architectural Form Definition Requirements in Annex A of this International Standard.

0.1 HyTime modules

The architectural forms and attributes of the HyTime language are grouped into five modules, each of which have both required and optional facilities. 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 object properties. The optional facilities provide lookup tables for commonly used elements, a mechanism for associating use and access policies with objects, and a mechanism for relating attributes and the content of elements to their semantic values by reference. The base module also defines the fundamental coordinate addressing notation used by all the other HyTime modules.

— 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 multiple locations is also possible. The syntax and semantics of these addressing mechanisms are independent of the data content notations of the data being addressed.

NOTE 1 The ability to resolve HyTime addresses in a given notation is dependent on software that can interpret that notation in terms of the abstractions HyTime uses for all addressing (see **6.1.1 Object representation**).

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

— Hyperlinks module

This module allows relationships (“hyperlinks”) to be established 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 2 The semantics of the modifiers are not defined by HyTime.

- 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 sizes of events in the target coordinate space.

In situations where the rendered position and size 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 sizes of projected events.

NOTE 3 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 object 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 objects, 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, or 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 architectural forms with one another, it can create its own architecture (for example, “MyArch”) defining its own set of architectural forms. These architectures may be derived wholly or in part from the HyTime architecture. The facilities for defining and using architectures are defined in annex A.3.

If, for example, a document is derived from the HyTime and MyArch architectures, 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-conforming 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 4 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 5 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.

NOTE 6 HyTime shares with the DSSSL standard (ISO/IEC 10179:1996, Document Style Semantics and Specification Language) the fundamental SGML property set and grove abstraction for representing and operating on parsed SGML documents (and other data objects for which groves can be constructed).

0.3 Organization of this International Standard

The structure of this International Standard 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 location address (clause 7), hyperlinks (clause 8), and scheduling (clause 9), clauses describe modules that are independent of one another.
- The rendition clause (clause 10) describes a module that is dependent on the scheduling module.
- The conformance clause (clause 11) 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 the SGML Extended Facilities, many of which are prerequisites for the other clauses.

- Annex B

This normative annex defines the HyTime property set.

- Annex C

This normative annex contains the complete HyTime and General Architecture meta-DTDs as they are used by architectural engines.

- Annex D

This informative annex identifies sources of supplementary tutorial and reference materials for 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 7 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 sizes 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 8 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 sizes 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 may determine 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/IEC 8824) and can be encoded according to the basic encoding rules of ISO/IEC 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 9 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 10 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 31-0:1992, *Quantities and units — Part 0: General principles*.

ISO 3166:1993, *Codes for the representation of names of countries*.

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*.

ISO/IEC 10179:1996, *Information technology — Processing languages — Document Style Semantics and Specification Language (DSSSL)*.

ISO/IEC 13673, *Information technology — Text and Office Systems — Conformance Testing for Standard Generalized Markup Language (SGML) Systems*¹⁾.

3 Definitions

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

3.1

anchor

an object (or list of objects) that is linked to other objects or lists of objects by a hyperlink

NOTE 11 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 12 An object is an anchor if and only if a hyperlink identifies it as such.

NOTE 13 An object could be identified as an anchor in several link ends, in the same or different hyperlinks.

3.2

anchloc

hyperlink anchor location address

3.3

application BOS

a bounded object set that is determined by an application according to its own rules

NOTE 14 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.

3.4

architectural forms

rules for creating and processing components of documents. There are four kinds: element form, attribute form, notation form, and data attribute form

¹⁾ To be published.

3.5

attribute form

an architectural form that applies to attributes of elements

3.6

auxiliary grove

a grove constructed by processing nodes in another grove

NOTE 15 For example, the data tokenizer grove constructed for use by a data location address is an auxiliary grove.

3.7

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

3.8

bounded object set

BOS

the subject that a HyTime application processes: a set of one or more documents and other information objects

NOTE 16 There are three kinds: HyTime BOS, application BOS, and effective BOS.

3.9

bounding region

the region of an FCS addressed by an fcsloc within which events are selected according to the selection precision specified for the fcsloc

3.10

children property

in groves, that subnode property of a node that is designated as the content property of the node. A node may have zero or one children properties

NOTE 17 If a content property is not nodal, then the node does not have a children property.

3.11

client architecture

an architecture derived from another architecture. The first architecture is said to be a client of the architecture from which it is derived

3.12

client document

a document conforming to an architecture. The document is said to be a client of the architecture to which it conforms

3.13

client DTD

the document type definition of a client document

3.14

clink

contextual link element form

3.15

content property

in groves, that property of a node that is designated as containing the semantic content of the node. A node may have zero or one content properties. A content property may be nodal or primitive. When a content property is nodal, the property is also the children property of the node

NOTE 18 For example, for SGML element nodes, the content property can contain elements, data characters, or other objects that occur syntactically within SGML elements, excluding objects that the rules of SGML exclude from the semantic content, such as ignored record ends and separators in element content.

3.16**content tree**

in groves, the tree formed by a node and the nodes in its children property

3.17**contextual hyperlink**

a hyperlink that occurs “in context”, meaning that one anchor of the link is the link element itself (a “self anchor”) and is a traversal initiation anchor. In an interactive application, the self anchor can be accessed externally from adjacent elements or data in the document hierarchy

NOTE 19 Any of the hyperlink element forms may be used contextually. The clink element form is always contextual.

3.18**contextual link element form**

an element form that represents a binary contextual hyperlink having the fixed anchor roles reference mark (refmark) for the self anchor and reference subject (refsub) for the other anchor

3.19**data attribute form**

an architectural form that applies to data attributes

3.20**data location address**

a location address that addresses the string and token data objects resulting from tokenization of character data

3.21**dataloc**

data location address

3.22**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

3.23**document**

a collection of information that is identified as a unit and that is intended for human perception

3.24**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 20 A document type definition can also include comments that describe the semantics of elements and attributes, and any application conventions.

3.25**effective BOS**

the bounded object set consisting of all the objects that at any given point have been successfully and fully integrated into the hyperdocument being processed

3.26**element form**

an architectural form that applies to elements

3.27**entity descriptor**

a component of an SDIF data stream that represents an external entity

3.28 entity tree

a tree structure whose nodes are entities, constructed by the following steps:

- 1) An SGML document entity or SGML subdocument entity is selected as the root node.
- 2) 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.
- 3) Step 2 is repeated for each parent node, until the leaves of the tree or a specified maximum number of levels is reached.

NOTE 21 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.

3.29 event

the occurrence of an object in a coordinate space. It associates the object with a scheduled extent

NOTE 22 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.

3.30 event projection

the conversion of the scheduled extent of an event from schedule to schedule (the “unprojected” to the “projected”)

NOTE 23 For example, from music time to real time, or from user device coordinates to real space units.

3.31 (scheduled) extent

size and position in a coordinate space. It consists of a dimension on each axis of the space

3.32 external identifier

a parameter of an SGML markup declaration (typically an entity declaration) that identifies an external information object

NOTE 24 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.

3.33 fcsloc

finite coordinate space location address

3.34 finite coordinate space location address

a location address that addresses events in an event schedule (or the objects scheduled by those events) by defining a bounding region enclosing the events selected

3.35 graph representation of property values

an abstract data structure consisting of a directed graph of nodes in which each node may be connected to other nodes by labeled arcs

3.36 grove

graph representation of property values

3.37 grove construction process

a process that constructs a primary or auxiliary grove

3.38**grove definition**

the combination of a grove plan, grove construction process, and grove source object

3.39**grove plan**

a specification of what modules, classes, and properties to include in a grove. Grove plans are used both to construct groves and to view existing groves

3.40**grove root**

the one node in a grove that does not have an origin

3.41**grove source**

the data (for primary groves) or nodes (for auxiliary groves) from which a grove is constructed

3.42**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 25 “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 26 It is possible that several HyTime documents, when designated as a hub document, could define the same HyTime BOS.

3.43**hyperdocument**

two or more documents or other information objects that are connected to one another by a web

NOTE 27 Access to a hyperdocument begins from a designated hub document.

NOTE 28 One hyperdocument may include another, otherwise independent hyperdocument, by declaring the hyperdocument’s hub document to be included as a “subhub”.

NOTE 29 A hyperdocument may represent a fixed or singular collection of data objects organized for a particular purpose (such as to represent common ownership or to support a particular rhetorical purpose) or it may be an arbitrary and transitory collection of objects. The HyTime bounded object set control facilities and activity policy association facilities, as well as application-specific mechanisms, can be used to define additional semantics for and constraints on the composition and use of hyperdocuments.

3.44**hyperlink**

an information structure that represents a relationship among two or more objects

NOTE 30 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 31 Hyperlinks can be assigned link types and names by applications and architectures that use HyTime.

NOTE 32 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.

3.45**hyperlink anchor location address**

a form of query location address that addresses objects by the anchor role names of the anchors of which the objects are members

3.46**hyperlink location address**

a form of query location address that addresses hyperlinks by link type

3.47**hypermedia application**

an information processing application that has hypertext and/or multimedia capabilities

NOTE 33 An implication of this definition is that “hypermedia” refers to the union of hypertext and multimedia, rather than their intersection.

NOTE 34 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.

3.48

hypermedia document

a document or hyperdocument that is used in a hypermedia application

NOTE 35 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.

3.49

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, in a variety of ways — but systems need support only the subset that is within their present capabilities

3.50

hytime attribute

an attribute whose definition is included in a HyTime architectural form

3.51

hytime BOS

a HyTime BOS is a tree of entities rooted at and specified (directly and/or indirectly) by the entity declarations and other specifications in a HyTime hub document

NOTE 36 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.

3.52

hytime document

an SGML document whose properties are represented essentially as defined in this International Standard

NOTE 37 A HyTime document is normally a conforming HyTime document (*see 11.1 Conforming HyTime document*).

3.53

hytime element

an instance of a HyTime element type

3.54

hytime element type

an element type in a HyTime document that conforms to a HyTime architectural form

NOTE 38 The element type itself is not defined by HyTime. In addition to attributes defined by the architectural form, it may have application-specific attributes.

3.55

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 39 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.

3.56

hytime hyperdocument

a hyperdocument whose hub document is a HyTime document

NOTE 40 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.

3.57

hytime system

an SGML system that includes a HyTime engine

3.58

hypertext

information that can be accessed in more than one order

NOTE 41 A hypertext can be a single document or a library of documents (a “hyperdocument”). For example:

- A novel is typically not designed to be 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.

3.59

initial referrer (to a location path)

a referrer to the first location step in a location path

NOTE 42 An initial referrer cannot be a location address because one location address that references another becomes a location step.

NOTE 43 Every step in a given location path has the same initial referrer.

3.60

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, in a variety of ways

3.61

link

depending on context, either a hyperlink or an SGML processing link (“link process”)

3.62

link end

in a hyperlink, the union of an anchor role name, the members of the anchor, and the other properties associated with the anchor, such as traversal rules

3.63

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 44 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.

3.64

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

3.65

linkloc

hyperlink location address

3.66

listloc

list location address

3.67

list location address

a location address that selects nodes from a node list by specifying one or more dimension specifications, one for each contiguous sequence of nodes selected

3.68

location ladder

a set of location addresses, known as “location rungs”, in which each rung is the location source of the rung below it

NOTE 45 A location ladder is visualized as running from top to bottom, from location source to path step. A rung that is a step in a location path is considered to be the bottom rung with respect to that path.

NOTE 46 A location ladder represents a progressive culling of the set of addressable objects as one proceeds downward (in other words, the top rung addresses the largest possible scope from which nodes may be selected by the rungs below it).

3.69

location address

an element form that represents the address of one or more objects. A reference to a location address is treated as a reference to the objects located by the location path that it begins

NOTE 47 A location address can be the location source of another location address, thereby forming a location ladder.

NOTE 48 A location address is the HyTime representation of what is commonly known in computing as an “indirect address”.

3.70

location path

a set of location addresses, known as “location steps”, in which the first step locates the second and so on. A location path can have branches, created when a step locates two or more objects, at least one of which is a location address. A branch terminates when its last step locates only objects that are not location addresses. The objects located by a location path are all those, other than steps, that are located by any of the steps in the path

NOTE 49 A location path is visualized as running from left to right, from initial referrer to objects addressed. Each step is the bottom rung of a location ladder.

NOTE 50 A location address can be a step in more than one location path, but can be the first step in only one.

3.71

location source

the set of objects from which a location address selects the objects that it addresses

NOTE 51 For example, the location source of a treeloc is the tree in which the node addressed by the treeloc is found.

3.72

mixedloc

mixed location address

a location address that addresses objects indirectly through its child location address elements

3.73

multimedia

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

3.74

(object) modification

the modification of an object by another object during rendition

NOTE 52 For example, routing audio signals through an effects box.

NOTE 53 HyTime deals with the scheduling and association of modifiers, but not the semantics of modification.

3.75

nameloc

named location address

3.76

named location address

a specialized form of mixed location address that addresses objects by their element IDs or entity names

3.77**named node list**

in groves, a node list in which all the member nodes exhibit a unique value for a common “name” property. The value of a name property may be either a string or a single node where the node is unique among all the nodes in the name values of the members of the node list

3.78**name-space location address**

a location address that addresses nodes in named node lists

3.79**nmsploc**

name-space location address

3.80**node**

in groves, an ordered set of properties representing a single object

3.81**notation form**

an architectural form that applies to data entities and notation data content

3.82**origin**

in groves, for a node, the node of which the node is a subnode. Every node in a grove, except the grove root, has exactly one origin node

3.83**patch**

an interconnection of modifiers

3.84**pathloc**

path location address

3.85**path location address**

a location address that addresses nodes in a tree by viewing the tree as a matrix where each column is the list of nodes from the root to a leaf

3.86**pelement**

pseudo-element

3.87**presentation**

a state of processing of a document in which it is ready for human perception

3.88**previous specified element**

the element in the SGML representation of the document that was most recently parsed

NOTE 54 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.

3.89**primary grove**

a grove constructed by processing source data

NOTE 55 For example, the grove constructed for an SGML document after the initial parse of the SGML source data is a primary grove.

3.90

principal tree

in groves, that content tree designated as being the principal tree by the grove's property set. In HyTime, the principal tree is the default implicit location source for tree location addresses

3.91

principal tree root

in groves, the node that is the root of the grove's principal tree. In the SGML property set, the document element is the principal tree root

3.92

projection

event projection

3.93

property location address

a location address that addresses the value of a property of a node in a grove

3.94

property set

a formal definition document, conforming to the requirements of the Property Set Definition Requirements in this International Standard, that specifies the node classes and properties to be used in constructing a grove

3.95

proploc

property location address

3.96

pseudo-element

in the SGML property set, an object whose children are derived from the data and other constructs occurring between a tag close and the next tag open. Pseudo-elements are children of elements

3.97

quantum

countable division of a coordinate axis

3.98

queryloc

query location address

3.99

query location address

a location address that uses a query to address nodes in a grove by their properties

3.100

real time

time in the every-day sense, as measured in seconds, minutes, hours, etc.

3.101

referent (of an ID reference)

an element whose ID is a value of an ID reference or ID reference list attribute (or content)

3.102

referential attribute

an attribute of an element that is designated as being a reference either by declaring it as an IDREF, IDREFS, ENTITY, or ENTITIES attribute or through the use of the HyTime refloc facility

NOTE 56 In the HyTime meta-DTD, referential attributes are identified by the conventional comment “— Reference —”.

NOTE 57 Content may also be designated as referential.

3.103**referrer (to a referent element)**

an element with a referential attribute (or content) that addresses the referent element

3.104**relative location address**

a location address that addresses nodes in a tree by specifying the genealogical relationship of the nodes to a starting node and then selecting nodes from the list of relatives

3.105**relloc**

relative location address

3.106**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 58 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.

3.107**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.

3.108**(location) rung**

an element in a location ladder

3.109**SDIF packer**

a program that creates an SDIF data stream

3.110**SDIF unpacker**

a program that decomposes an SDIF data stream into its constituent entities

NOTE 59 If necessary, the SDIF unpacker will modify the system identifier parameter of markup declarations to be consistent with storage addresses in its environment.

3.111**self anchor**

a hyperlink that addresses itself as one or more of its anchors

3.112**semantic grove**

a grove created by an application or architecture engine. The nodes of a semantic grove reflect the data structures required to implement the application- or architecture-specific semantics

NOTE 60 The HyTime architecture's semantic grove is defined by the HyTime property set (see Annex B).

3.113**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 61 When SDIF is used as a hyperdocument interchange format for HyTime, the "main" document, if there is more than one, is the hub document.

3.114

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

3.115

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

3.116

subhub

a hub document declared in another hub document and whose own HyTime BOS is added to the parent document’s HyTime BOS

3.117

SMU name

a name, declared by a notation declaration, that identifies a standard measurement unit

3.118

(location) step

an element in a location path

3.119

subnode

in groves, a node whose origin node exhibits a property of which the node is a member

3.120

subnode property

in groves, a property whose value must consist of nodes whose origin is the node exhibiting the property

3.121

subnode tree

the tree of nodes consisting of a node and all of its subnodes

3.122

target (of an ID reference)

the referent or, if that is a location address, the objects located by the location path that the referent begins

3.123

treeloc

tree location address

3.124

tree location address

a location address that addresses a single node of a tree in the classical manner by selecting a node from an addressable range at each level of the tree, starting at the root

3.125

validating HyTime engine

a conforming HyTime engine that can find and report a reportable HyTime error if (and only if) one exists

3.126

view

a presentation of a web and the anchors that it connects

3.127

web

a set of one or more hyperlinks that are used together

NOTE 62 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.
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

HyTime is an enabling document architecture whose definition conforms to the Architectural Form Definition Requirements in annex A.3 of this International Standard.

NOTE 63 That annex should be read in conjunction with this clause.

The specification of HyTime is accomplished by a combination of narrative text and formal definitions.²⁾ The formal definitions are expressed in SGML.³⁾

NOTE 64 This International Standard uses editorial conventions mandated by 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 International Standard, can be found in Annex D. However, the specific conventions for using SGML to define HyTime architectural forms are described in annex A.3.

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 65 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. Complete meta-DTDs are found in Annex C.

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 are presented, as follows:

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

²⁾ This International Standard 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.

³⁾ 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 may be copied as specified in the following copyright notice: (C) International Organization for Standardization 1992, 1997. 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.

The attribute 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 forms) the description is not repeated.

The declarations include comments, called “conventional comments”, that follow conventions established in this clause to specify syntactic and semantic constraints and other information that is known to the HyTime engine. The conventional comments do not extend SGML in any way. They are used in the architecture definitions only, as a notation for the documentation of the architecture. They need not be included in application DTDs and, if they are included, the SGML parser will treat them as it would any other comment.

5.1 RCS name, full name, description, and clause

Every form name is followed by comments giving the full name, a description of the form, and the number of the clause in this International Standard in which the form is defined.

Attribute forms that are used by more than one element or notation form are given names in the reference concrete syntax for ease of reference. The RCS name is given in a comment immediately following the ATTLIST keyword of the attribute list declaration. Attribute lists that apply to a single element or notation form do not have this comment.

Individual attributes have a full name and description comment.

5.2 Lexical type

The lexical types of attributes and data content are specified by conventional comments beginning with the word “Lextype” or “Ulextype”, followed by a lexical model in the “HyTime lexical model notation” (HyLex) [see **A.2.3 HyTime lexical model notation (HyLex)**]. The word “Lextype” signifies a normalized lexical model, while “Ulextype” signifies an unnormalized one. The lexical type names used in conventional comments are defined in the HyTime lexical type set (see **C.1 HyTime Lexical Types**).

Names matching ATTNAM tokens in lexical types are searched for first in the client element or entity, followed by each architectural element or entity in the chain from the client element or entity to and including the architectural element or entity to which the ATTNAM lexttype applies.

Names matching NOTATION tokens in lexical types are searched for first in the client document, followed by each architectural document in the chain from the client document to and including the architectural document to which the NOTATION lexttype applies.

NOTE 66 HyLex is not only used to document architectures. It is part of the General Architecture and can therefore be used by application designers as well. However, use of HyLex in conventional comments in this International Standard does not require that implementations support its use by applications.

5.3 Constraints

Comments labeled “Constraints” define additional semantic or syntactic constraints on the constructs they follow. Constraint comments that follow the name of a form define constraints on the use of that form in general. Constraint comments that follow a component of a declaration (for example, the default value prescription of an attribute declaration), define the specific constraints on that component.

5.4 Note

Note comments provide additional information not provided by the other comment types and are informational rather than constraining.

5.5 Associated attribute forms and attribute lists

Within its declaration, an element form or notation form identifies its associated attribute forms by means of conventional comments consisting of any of the words “Attributes”, “OptionalAttributes”, or “CommonAttributes”; followed by the names of the attribute forms.

The “Attributes” comment lists attribute forms that are always associated with the form. The “OptionalAttributes” comment lists attribute forms that are associated with optional facilities that are not themselves required by the form; these attribute forms will only be provided by the form when the relevant optional facilities are supported. The “CommonAttributes” comment lists the common attribute forms of the architecture as a reminder of their availability.

5.6 Referrers

The “Referrers” comment lists those element forms that may or are required to refer to this form. The comment is a list of element forms, followed by the name of the attribute by which the reference is made unless the reference is made in element content.

5.7 Conventions for attribute form declarations

Declared values that require ID references are specified with either the SGML IDREF or IDREFS keywords, or with ID references indicated by a lextype conventional comment. All attributes that are semantically referential, regardless of declared value, are so indicated by the use of a conventional comment consisting of the word “Reference”.

If a referential attribute definition contains a conventional comment containing the word “Reftype” followed by a name or SGML OR group, the target or targets of the reference must be elements that conform to the named element form or an element form whose name occurs in the OR group. Note that the OR group is meta: the names are architectural forms rather than element types.

NOTE 67 The reftype conventional comment should not be confused with the more powerful reftype facilities of the General Architecture and HyTime, which are for use by application designers. Use of the reftype comment in an architecture definition does not require the architecture to support the General Architecture or HyTime reftype facilities.

An attribute’s default value is documented in the default value prescription of the attribute definition, except that for impliable and content-reference attributes it is documented in a conventional comment that begins with “Default”.

If the attribute definition contains a conventional comment that consists of the word “Constant”, it signifies that the attribute is a constant attribute; that is, the attribute must have the same value for all elements of a given element type if it has a value for any element of that type.

NOTE 68 In SGML, a fixed attribute is one whose specified value (if any) must always be equal to its default. A fixed attribute is necessarily a constant attribute, but the converse is not necessarily true.

NOTE 69 If an attribute is constant in the meta-DTD, an application should declare it as a fixed attribute in the DTD, but this is not required because an architecture only defines conformance for document instances. However, a validating HyTime or General Architecture engine can optionally warn when this is not the case.

The “Constant”, “Lextype”, “Ulextype”, “Reference”, and “Reftype” conventional comments could be recognized and processed to provide validation beyond that supported by SGML alone.

5.8 Identification of optional facilities

HyTime’s facilities are organized into modules, each of which is documented in a separate clause of this International Standard. All modules except the base module are optional and most facilities of each module are optional. Whether or not a facility is optional is indicated by the use of parameter entities to control marked sections in the meta-DTD. For each optional facility there is a parameter entity whose name is the same as the name of the optional facility. These parameter entities are set to “INCLUDE” by naming them in one of the architectural facility control attributes defined for the HyTime notation (see **6.3 HyTime support declarations**).

The total set of options is summarized in the integrated HyTime meta-DTD provided in *C (normative) Architectural Meta-Declarations*.

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 source object representation in a HyTime document. Groves are used for the representation of abstract objects derived from source objects.

NOTE 70 While the source of HyTime documents is represented in SGML, all HyTime semantics are defined in terms of operations on nodes in groves, e.g., on the abstract data objects described by SGML documents (or any other type of data for which a suitable grove representation can be provided).

This sub-clause describes the key SGML constructs on which HyTime relies.

NOTE 71 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 owner of a chapter or revision date of a table. Many element types have an <i>ID</i> 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 <i>ID reference</i> attribute whose value is the label of the element referenced (as defined by that element’s ID attribute). |
| entity | A unit of virtual information storage that contains part of a document (sometimes all of one or even more than 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 Entity structure). |

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 (typically by redefining entities that are referenced in the DTD). This technique provides the flexibility necessary to accommodate a wide variety of applications.

Processors that act on the semantic objects and their data content do not operate directly on the source representation. Instead, they must first create an abstract, “in-memory” representation of the structures and data in the source. In the case of SGML, an SGML parser processes the source document to recognize markup and data and provides information about what it found to a processing application. The processing application then operates on its particular abstraction derived from the result of parsing the document.

The HyTime standard uses a standard form of abstract structure representation called “groves”. All HyTime-defined processing operates on groves (rather than on the unparsed source data from which groves are constructed). Groves consist of “nodes”, which exhibit “properties”. Each node is of a particular node class. The node classes and their properties are defined in “property sets”.

The combination of property sets and groves provides a standard for the representation of application-specific data views of SGML documents (and, potentially, other data notations) and enables interoperation of applications by, at a minimum, providing a common design formalism by which applications can define the data objects on which they operate and the results of those operations, irrespective of how they actually represent or process data objects internally. In other words, groves and property sets make it possible to discuss the processing of structured data without first defining the implementation of the processors themselves. Groves are discussed in detail in **7.1.4 Groves and Location Addressing** and in **A.4 Property Set Definition Requirements (PSDR)**.

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 72 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 informally 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 73 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 7).

6.1.2 Object identification and addressing

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

A hyperlink may address its anchors by names that are unique within some name space or by position within a list or tree. Time synchronization uses coordinate addresses of events on a time axis: the address of event A may be expressed in terms of its position and size on the axis, or 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.

In HyTime, addressing is applied to nodes in groves and the result of any location address is the list of nodes addressed. Syntactically, the list of nodes addressed can be given a unique name by specifying a unique ID for the location address element that addresses the nodes. Whether it has a name or not, every such node list is uniquely addressable because it becomes the property of a node in the “HyTime semantic grove”. The nodes exhibiting these properties can then be addressed by name, if they have one, or by other node addressing methods, if supported.

NOTE 74 If an application only supports ID-based addressing, all elements on which other elements depend and use by reference must have IDs.

6.1.2.1 Name space addressing

In the grove model, any node class may exhibit a “name” property that uniquely identifies it within a “name space”. In groves, a name space is a property whose value is a list of nodes, all of which share the same “name” property and each of which exhibits a unique value for its name. SGML defines two primary name spaces that are made directly accessible through the syntax of SGML documents: application-defined information components, called elements, and 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). 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.

Conceptually, all location addresses begin as ID references.

NOTE 75 HyTime also provides “shortcut” forms of address in which the ID reference or references are implicit. For example, when coordinate addressing is supported, the reftype facility allows the direct use of coordinate addresses in place of ID references. However, for all such shortcuts, the equivalent reference can be constructed using ID references and other fundamental forms of addressing.

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) in order to address the entities that contain those locations.

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 of interest) and a specific number of subsequent contiguous quanta for each of the axes of the coordinate space. This combination of position and size is called an extent.

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

When the location address module is supported, “location address” elements can be defined that associate an ID (directly or indirectly) 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 76 For example, “the third word in the sentence”.

It also allows references to arbitrary portions of an object.

NOTE 77 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 query against its properties. Nodes in any grove, regardless of the data notation from which they were derived, are addressable by HyTime location addresses. Because the properties of nodes in groves are formally defined in a property set definition, it is possible to have general query facilities that can operate on any grove. Groves can, potentially, be constructed from any data notation by a “grove construction process” that understands the syntax and semantics of the notation (however the degree to which a grove can fully express the structure or content of a given notation is dependent on its similarity to SGML — groves are not intended nor guaranteed to be capable of fully representing any possible notation).

NOTE 78 In practical terms, “constructing a grove” may simply mean providing an interface that takes as input a HyTime location address and returns the data that would have been addressed had a literal grove been constructed. In other words, there is no requirement to literally build a grove; it is only necessary to behave as if one had been built.

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 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** Entity structure).

NOTE 79 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. For objects that have an internal structure, those structures will only be addressable through HyTime facilities if a grove is constructed for them. An object for which there is no grove can only be addressed by reference to an entity node in an SGML document grove that declares the object as a data entity.

6.2.2 Object identification and addressing

Object identification facilities within HyTime documents are provided by SGML and are defined in ISO 8879. Public identifiers of external documents and information objects must conform to ISO 9070.

NOTE 80 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 81 Forms of public identifier not supported by ISO 9070, such as distributed object references with semantic connotations, can be treated as formal system identifiers [see **A.6** Formal System Identifier Definition Requirements (FSIDR)].

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.

NOTE 82 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 Bounded object set (BOS)

Several HyTime semantics, including hyperlink anchor semantics, activity policy semantics, and scheduling semantics, are applied to referents by referencing constructs (referrers). The referent of any referrer may be in a different object than the referrer. Since no system can be expected to process all of the documents in existence to determine what semantics may have been applied to any given piece of information, it is necessary to specify which documents contain all the referrers that are needed in order to understand the referents in a given HyTime hyperdocument.

Also, as a HyTime hyperdocument may consist of more than one object, authors must be able to specify what objects need to be present in order to have a complete copy of the hyperdocument.

For these reasons, the number of documents and other information objects that a HyTime system can be expected to regard as a hyperdocument must be finite — a “bounded object set” (BOS). If the exact set of required objects is not known, there is a potential for incorporating vast amounts of unwanted information and incurring unnecessary delay, expense and inconvenience. HyTime provides facilities for managing bounded object sets. These facilities consist of common data attributes, attributes of the HyDoc element form, and the BOS specification element form (see 6.5.2 HyTime BOS control data attributes and 6.4 HyTime document).

Conceptually speaking, there are three kinds of BOS:

HyTime BOS Every HyTime hyperdocument has exactly one “hub” document; processing of the entire HyTime hyperdocument nominally begins with the hub, which is nominally the HyTime document through which entry to the HyTime hyperdocument was gained, that is, the HyTime document with which the processing session begins. The HyTime BOS is the BOS specified (directly and indirectly) by a HyTime hub document. Therefore, the author of a HyTime hyperdocument has control of the set of objects considered to be members of its HyTime BOS. The membership of objects in the HyTime BOS is determined according to rules defined by this International Standard regarding the semantics of the boslevel, inbos, and subhub common data attributes, the semantics of the maxbos, boslevel, and bosspec attributes of the HyDoc architectural form, and the semantics of the boslevel and inbos attributes of the bosspec architectural form. The objects in a HyTime BOS are also classifiable by HyTime hyperdocument authors as having foreground or background priority.

application BOS The application BOS is the BOS intended by the software programmer of a HyTime system to constitute the hyperdocument.

NOTE 83 The concept of the application BOS includes an implicit recognition of the fact that some systems may be deliberately designed to process hyperdocuments which differ, in terms of the set of member objects used, from the HyTime BOS. For example, there may be systems that regard all documents referenced by World Wide Web-style URLs that appear in the hub document to be in the BOS, even when the documents referenced by such URLs are not specified by any corresponding entity declarations anywhere in the hyperdocument.

effective BOS NOTE 84 If no value is specified for the boslevel attribute of the hub document’s HyDoc document element, there is no HyTime BOS and, of necessity, the application BOS governs the assembly of the effective BOS. The BOS consisting of all the objects that at any given point have been successfully and fully integrated into the hyperdocument being processed.

NOTE 85 The concept of the effective BOS is an implicit recognition of the fact that some objects that the hyperdocument author regarded as necessary (and which are therefore in the HyTime BOS), and/or which the system programmer regarded as necessary (and which are therefore in the application BOS), may not be retrievable or integratable into the hyperdocument with complete success. Much of this International Standard is primarily concerned with the semantics of a set of objects that has been fully and successfully integrated into a hyperdocument, and therefore it is primarily concerned with the effective BOS, irrespective of how the membership in the effective BOS was determined, or which members of the HyTime and/or application BOSs had to be left out or exceptionally included for any reason.

As the HyTime BOS is the only BOS whose membership is defined by this International Standard, and as the HyTime BOS is the only standardized indication of what objects an author intends the hyperdocument to consist, there is reason to expect that, in many applications and instances in which HyTime hyperdocuments are processed, the application BOS and the HyTime BOS will be identical, and, therefore, barring mishap, the effective BOS will be the HyTime BOS.

This International Standard does not provide means for expressing application BOSs or effective BOSs; it provides such means only for HyTime BOSs.

NOTE 86 Of course, nothing prevents the expression of an application BOS or an actual effective BOS as a HyTime BOS in a HyTime document.

Except possibly for addressing information needed to refer to portions of the content of documents that are not present in the effective BOS, a HyTime system is not required to have any knowledge of the content of documents and objects that are not in the effective BOS. If a HyTime system obtains access to the content of such documents, such as when resolving the addresses of anchors within such documents, such documents must first be added to the effective BOS. Insofar as HyTime processing semantics and the system's requirements dictate, a HyTime system maintains knowledge of and access to all the referents within the effective BOS of all referrers within the effective BOS.

NOTE 87 In other words, the effective BOS is the set of information objects for which a HyTime system has actually assumed responsibility at any point in time. The effective BOS concept — the idea that a system has only a finite number of objects for which it is responsible — makes references with bidirectional semantics from and to external documents both practical and scalable.

For example, if the scheduling module is supported, the effective BOS contains all the fcs elements corresponding to all the FCSs for which the system is responsible, and all the scheduling constructs known to affect them. Even if an fcs element exists in the effective BOS, any schedules, batons, or wands that are directly scheduled or projected onto the logical FCS to which it corresponds will have no effect unless they, too, are included in the effective BOS. (In other words, the system is not required to know about anything affecting FCSs that is not in the effective BOS.) Schedules, batons, and wands that only affect FCSs that correspond to fcs elements that are not present in the effective BOS have no effect unless and until the documents in which such fcs elements are declared are added to the effective BOS. (Systems may support the addition of documents containing such referenced fcs elements to the effective BOS even when they are not members of the HyTime BOS; such added objects are, by definition, members of the application BOS.)

For another example, if the activity option of the base module is supported, the effective BOS determines the effectivity of activity policies. Actrules in the effective BOS are in effect for the objects in the effective BOS that they govern, according to the policies that are also in the effective BOS.

For a third example, in the case of hyperlinks, the system knows all the anchors within the effective BOS of all the links within the effective BOS. It need not know the anchors of any links if those anchors are not in the effective BOS, and it need not know any anchors that are the anchors of links that are not in the effective BOS. If a link is traversed to an anchor outside the effective BOS, the object containing that anchor must first be added to the effective BOS. Some systems may do this without any special user interaction; others may warn users that, for example, the effective BOS will thereafter contain an object that was not specified in the HyTime BOS.

NOTE 88 Adding a document to the effective BOS does not necessarily mean adding that document's HyTime BOS to the effective BOS, as if the added document were itself the hub document. Only the hub document specifies how the HyTime BOS is to be derived, and, in any case, some systems may ignore even the HyTime BOS.

A HyTime bounded object set ("HyTime BOS") can be determined automatically by a HyTime system by "discovering" an "entity tree", using the SGML document entity of the hub document as the root. The entity tree includes external document, subdocument, and data 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 discovered. Therefore, if external references are made from data content using a notation-specific form of reference, such entities will not be included in the HyTime bounded object set unless the document originator also creates entity declarations for them.

NOTE 89 This International Standard does not require systems to assemble BOSs according to any particular methodology or order. However, purely as an example, it may be useful to consider the following methodology for determining the membership of objects in a HyTime BOS. (Considerable optimization of the performance of interactive HyTime hyperdocument applications can be achieved by judicious use of foreground and background priority. For clarity, the following methodology does not take the possibility of background priority into account.)

1) Determine the unadjusted HyTime BOS.

a) For each entity declared in the hub document entity whose subhub attribute's value is subhub, complete a HyTime BOS assembly process exactly like the one being done for the hub document. This should be done recursively if subhubs declare entities as subhubs. The HyTime BOS of each subhub document entity is added to the tentative HyTime BOS of its parent document entity. Do not regard as a subhub any entity whose parent is not a hub or subhub, even if its parent declares it using a subhub attribute with a value of "subhub".

b) Discover the entity tree of which the SGML hub (or subhub) document entity is the root, adding each entity to the tentative HyTime BOS. Ignore all entities whose subhub attribute's value is "subhub" unless their parent entity was neither a hub nor a subhub, as they have already been taken into account in the previous step. During the entity tree discovery process, do not exceed the number of recursions specified by the smaller of (i) the value of the maxbos attribute of the hub's (or subhub's) document element's maxbos attribute, or (ii) the value of the boslevel common data attribute (which may have a default value specified by the boslevel attribute of the hub's [or subhub's] document element's boslevel attribute) of the corresponding entity declaration in the hub document entity. If, during the entity tree discovery process, an entity declaration's inbos attribute has a value of "inbos", include it, too, in the tentative HyTime BOS even if doing so exceeds the maxbos or relevant boslevel specification.

2) Apply all adjustments specified by bosspecs referenced by the bosspec attribute of the document element of the hub (or subhub) document whose inbos attribute's value, if any, is not "notinbos". This may require additional entity tree discovery processing. If the value of the inbos attribute of a bosspec is "inbos", add all the objects addressed by the content and boslevel attribute of the bosspec to the tentative HyTime BOS.

3) Apply all adjustments specified by bosspecs referenced by the bosspec attribute of the document element of the hub (or subhub) document whose inbos attribute's value is "notinbos". Whatever objects remain constitute the HyTime BOS.

Limits can be placed on the depth of the HyTime BOS entity tree discovery process by the maxbos attribute of the hub document's document element, by the boslevel common data attribute on entity declarations, and by bosspec elements referenced by the bosspec attribute of the hub document's document element.

The HyTime BOS, if specified, represents the author's intentions with respect to the objects that participate in the hyperdocument. However, a system may use either an application BOS or a HyTime BOS to guide the assembly of its effective BOS. Moreover, it is possible for systems to exclude, voluntarily or involuntarily, some of the member objects of the HyTime BOS from the effective BOS, and/or to include objects in the effective BOS that are not members of the HyTime BOS. One possibility is for the system to use the HyTime BOS as a starting effective BOS, but to allow entities to be added and deleted by the user. (By definition, once the effective BOS is no longer the same as the HyTime BOS, the effective BOS is an application BOS.) Therefore, the question of whether the system is presenting the hyperdocument as the author intended at any given point is an information accuracy issue.

It is possible for the effective BOS to be neither an application BOS nor a HyTime BOS, if the intended HyTime or application BOS cannot be fully assembled due to environmental processing constraints, such as the lack of a network connection. It is also possible for a system to regard a document other than the session-start document as a hub document, and to use the HyTime BOS that it specifies in some fashion. All three types of BOS (HyTime, application, and effective) always contain, at minimum, a hub document.

6.2.5 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 90 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.

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.

An SDIF data stream can carry any bounded object set, and is not strictly limited to HyTime 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.

6.2.5.1 SDIF packer

The process (application or HyTime engine) that determines the BOS submits the names of the entities that comprise 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, the SDIF packer 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 91 The SDIF "packer" process that creates the data stream is aware of the entity declarations, and therefore of the notations (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 **A.6.3 Containers**).

6.2.5.2 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 (because, e.g., 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 92 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 support declarations

A document may indicate conformance to HyTime by following the procedures in **A.3.3.1 Enabling architecture use of APPINFO parameter**. That is, the APPINFO parameter of the SGML declaration declares the existence of an ArcBase declaration, which identifies the HyTime architecture support declarations.

NOTE 93 The use of APPINFO serves to declare the use of architectures and HyTime within SGML declarations. However, it is sufficient to enable architectural processing to declare the use of HyTime with the ArcBase processing instruction within DOCTYPE declarations.

The support declarations include support attributes that are defined for all architectures. They are discussed in **A.3.4 Architecture support declarations**.

The support attributes for HyTime include support attributes for each module that control the use of optional facilities that affect the syntax of the HyTime meta-DTD. Additional support attributes control the use of facilities that have semantic, but not syntactic, implications.

There is a facility support attribute named after each module. The attribute value contains the names of the module's optional facilities for which support is being declared. If only the module name is specified, then only support of the module's mandatory facilities is declared, if any (some modules have no mandatory facilities). If other facilities are specified then support for the module is implied and need not be specified. If the specified attribute value is the empty string ("") or the attribute is not specified at all and the default value declared in the notation attribute declaration list is "#IMPLIED", then the module is not declared at all.

NOTE 94 For example, the following attributes declare that support is required for the optional descxt facility of the base module and the non-optional facilities of the sched module. The links and rendition modules are not required at all.

base="descxt" sched="sched"

Some options require qualification in addition to stating their support. For these options, separate attributes are provided where the attribute name is the name of the option, for example:

manyaxes="3"

When an option qualification attribute sets limits, if no value is specified, there is no limit for that option. Option-specific attributes are ignored if the option to which they apply is not supported.

The attribute **highest quantum count limit** (*hyqcnt*) declares a limit that no quantum count in the document instance will exceed, including the FCS dimensions of coordinate axes and the number of nodes in node lists. The hyqcnt is specified as a power of two that exceeds the limit. The HyTime default (and the minimum for any use of HyTime) is 4,294,967,295.

NOTE 95 In other words, when the default hyqcnt is used, a HyTime implementation is never required to accumulate a quantum count greater than the capacity of an unsigned 32-bit integer: $(2^{32}-1)$.

An application designer is required to choose granularities that will not cause this limit to be exceeded.

NOTE 96 The hyqcnt gives the application designer wide latitude in specifying the precision of HyTime coordinate addressing. For this reason, HyTime's use of finite quanta rather than infinite points does not restrict HyTime applications in any practical way.

The attribute **any SGML declaration allowed** (*anysgml*) specifies whether (*anysgml*) or not (*nanysgml*) the HyTime processor allows members of a hyperdocument to use different SGML declarations. When the value is "nanysgml", all SGML documents in the hyperdocument must conform to the same SGML declaration.

The attribute **external references allowed** (*exrefs*) specifies whether (*exrefs*) or not (*nexrefs*) external references are allowed. When external references are not allowed, all location addressing (except direct entity reference) must be to objects within the document from which the reference is made. The default is to allow external references. This option is ignored if the location address module is not supported.

The attribute **SGML model groups for reftype** (*refmodel*) specifies whether (*SGMLmdl*) or not (*nSGMLmdl*) reftype attributes may specify any valid SGML content model or are limited to repeating OR groups of element type names.

The attribute **maximum number of anchors allowed in hyperlinks** (*manyanch*) specifies the maximum number of anchors allowed for hyperlink instances. If the value is unspecified or is zero, the number of anchors is unlimited. This option is ignored if the hyperlinks module is not supported.

The attribute **maximum number of axes allowed in coordinate spaces** (*manyaxes*) specifies the maximum number of axes allowed for finite coordinate spaces. If the value is unspecified or is zero, the number of axes is unlimited. This option is ignored if the scheduling module is not supported.

The complete set of facility attributes, with all options supported, is:

```
<! -- HyTime Support Declarations -->
<? IS10744 ArcBase HyTime>
<! NOTATION
  HyTime          -- HyTime Architecture --
                  -- A base architecture used in conformance with the Architectural Form Definition Requirements of
                    International Standard ISO/IEC 10744. --
  PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR ARCBASE
          Hypermedia/Time-based Structuring Language (HyTime)//EN"
>
<! ATTLIST #NOTATION HyTime
  ArcFormA  NAME  HyTime
  ArcNamrA  NAME  HyNames
  ArcSuprA  NAME  sHyTime
  ArcIgnDA  NAME  HyIgnD
  ArcDocF   NAME  #FIXED HyDoc
  ArcDTD    CDATA "HyTime"
          -- NAMELEN must be at least 9 because the HyTime meta-DTD uses 8-character parameter entity names. --
  ArcQuant  CDATA #FIXED "NAMELEN 9"

  ArcDataF  NAME  #FIXED HyBridN
  ArcBridF  NAME  #FIXED HyBrid

  -- For some HyTime forms, the generic identifiers of client elements are meaningful to a HyTime engine and must be
  customized. Therefore, automatic form mapping may be inappropriate. --

ArcAuto (ArcAuto|nArcAuto) nArcAuto
ArcOptSA NAMES      "GenArc base links locs rend sched"
GenArc              -- General architecture facilities --
  CDATA              -- Lextype: csname+ --
  "altreps dafv dvlst HyLex HyOrd included ireftype lexttype opacity REGEX superdcn"
base                -- Base module facilities --
  CDATA              -- Lextype: csname+ --
  "activity actypes bos bosspec conloc descxtxt dmspec HyDimLst HyDimSpc HyFunk markfun valueref"
locs                -- Location address module facilities --
                  -- Clause: 6.3 --
  CDATA              -- Lextype: csname+ --
  "agrovdef bibloc dataloc datatok grovplan listloc mixedloc multloc nameloc nmsploc pathloc pgrovdef proploc queryloc refetl
  referatt refloc reftype relloc spanloc treecom treeloc treetype"
links               -- Hyperlinks module facilities --
                  -- Clause: 6.3 --
  CDATA              -- Lextype: csname+ --
  "agglink anchloc clink hylink ilink linkloc traverse varlink"
sched               -- Scheduling module facilities --
                  -- Clause: 6.3 --
  CDATA              -- Lextype: csname+ --
  "calibrat calspec dimref fcsloc grpdex grppepet HyCalSpc HyExSpec HyExtLst HyGrand measure objalign pulsemap sched"
```

```

rend                -- Rendition module facilities --
                    -- Clause: 6.3 --
    CDATA            -- Lextype: csname+ --
    "baton batonseq HyDimPro HyExPro HyPro modify obextent patch project proseq wand wndpatch"
anysgml              -- Any SGML declaration allowed --
                    -- Clause: 6.3 --
    (anysgml | nanysgml)
    nanysgml
exrefs               -- External references allowed --
                    -- Clause: 6.3 --
    (exrefs | nexrefs)
    exrefs
refmodel             -- SGML model groups for reftyp --
                    -- Clause: 6.3 --
    (SGMLmd1 | nSGMLmd1)
    nSGMLmd1
hyqcnt               -- Highest quantum count limit --
                    -- Clause: 6.3 --
    NUMBER 32        -- Constraint: power of 2 >= 32 --
manyanch             -- Maximum number of anchors allowed in hyperlinks --
                    -- Clause: 6.3 --
    NUMBER            -- Constraint: must be >= 2 --
    #IMPLIED          -- Default: no limit --
manyaxes             -- Maximum number of axes allowed in coordinate
                    spaces --
                    -- Clause: 6.3 --
    NUMBER            -- Constraint: must be >= 1 --
    #IMPLIED          -- Default: no limit --
>
<! NOTATION AFDRMeta
    PUBLIC "ISO/IEC 10744:1997//NOTATION AFDR Meta-DTD Notation//EN"
>
<! ENTITY
    HyTime            -- HyTime meta-DTD --
                    -- Clause: 6.3 --
    PUBLIC "ISO/IEC 10744:1997//DTD AFDR Meta-DTD
        Hypermedia/Time-based Structuring Language (HyTime)//EN"
    CDATA AFDRMeta
>

```

6.4 HyTime document

The element form **HyTime document** (*HyDoc*) is an architectural document element form (ArcDocF) (see **A.3.6.1 Architectural document element**).

```

<!-- Content model parameter entities -->
<! entity %
    HyCFC             -- HyTime context-free content --
                    -- Note: % loc, % link, % resorce qualify but are used as meta-inclusions rather than
                    meta-proper-subelements --
    "% GACFC; | HyBrid | fcs"
>
<! entity %

```

```

    loc                -- Location address forms --
    "anchloc|bibloc|dataloc|fcsloc|linkloc|listloc|mixedloc|nameloc|nmsploc|pathloc|proploc|queryloc|relloc|treeloc"
>
<! entity %
    link              -- Hyperlink forms --
    "agglink|clink|hylink|ilink|varlink"
>
<! entity %
    resbase           -- Base module resource forms --
    "actrule|bosspec|desctab"
>
<! entity %
    resloc            -- Location address module resource forms --
    "agrovdef|datatok|groplan|ipgrovdef"
>
<! entity %
    resschd           -- Scheduling module resource forms --
    "calspec|evsched|extent|extlist|measure"
>
<! entity %
    resrend           -- Rendition module resource forms --
    "baton|batrule|batseq|modpatch|modrule|projectr|prorule|proseq|rendrule|wandrule|wand|wndpatch"
>
<! entity %
    resorce           -- All resource architectural forms --
    "% resbase;| %resloc;| %resschd;| % resrend;"
>
<! element
    HyDoc             -- HyTime document element --
                    -- Clause: 6.4 --

    - O
    (% HyCFC;)*
    + (% link;| % loc;| % resorce;)
-- OptionalAttributes [base]: bos, bosspecat --
-- OptionalAttributes [locs]: dgrvplan --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmodl, ireftype, lexttype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
>

```

6.5 HyTime Bounded object set

Support for the management of HyTime bounded object sets is managed through the HyTime bos and bosspecat attribute forms of the HyDoc element form, the bosdatt data attribute form, and the bosspec element form.

6.5.1 HyTime bounded object set attributes

The attributes of the **HyTime bounded object set** (*bos*) attribute form control the construction of HyTime bounded object sets when the HyDoc elements that exhibit them are the document elements of HyTime hub documents.

Conceptually speaking, when determining which objects are members of the HyTime BOS of a HyTime hyperdocument, a HyTime application uses an “entity tree discovery” process in which entities that are recursively declared are examined and reported as being candidates for membership in the HyTime BOS [see **6.2.4 Bounded object set (BOS)**]. The attribute **maximum bounded object set level** (*maxbos*) specifies the nominal maximum number of recursions permitted in the normal entity tree discovery process. The value of the maxbos attribute applies to all entities declared in the hub document except subhub document entities (see **6.5.2 HyTime BOS control data attributes**).

NOTE 97 The value of maxbos is said to specify the “nominal” number of recursions because it may be overridden by the inbos common data attribute, by the inclusion of subhubs, and by the bosspec attribute of the HyDoc element.

A maxbos value of 0 means that the number of recursions to be made in the course of the normal entity tree discovery process is not limited. A maxbos value of 1 means that only the hub document has nominal membership in the HyTime BOS.

NOTE 98 In other words, if maxbos is 1, the normal entity tree discovery process does not recurse at all.

A maxbos value of 2 means that the normal entity tree discovery process should stop after one recursion, that is, all of the entities declared in the hub document are nominally members of the HyTime BOS, but there should be no examination of the entities declared in the entities declared in the hub document. A maxbos value of 3 means there is a limit of two recursions, and so on.

The maxbos value takes precedence over the boslevel attribute of a data entity when the data entity’s boslevel would result in a BOS larger than that allowed by the maxbos level.

NOTE 99 In other words, the nominal maximum number of recursions in the normal entity tree discovery process whose root is an entity declared in the hub document is actually limited to the smaller of the value of the maxbos attribute of the hub document’s HyDoc element and the value of the boslevel common data attribute of such entity.

The attribute **bounded object set level** (*sboslevel*) specifies the default bounding level to be used for data entities that do not specify a boslevel data attribute. The semantics of the numbers used in its value are the same as those of the maxbos attribute. If no value is specified for the boslevel attribute, there is no HyTime BOS; the author of the hyperdocument is, in effect, delegating responsibility to the system and asking that the application BOS govern the assembly of the effective BOS [see **6.2.4 bounded object set (BOS)**].

```
<![ % bos; [
< ! attlist
    -- bos --      HyTime bounded object set --
                    -- Clause: 6.5.1 --

(HyDoc)
maxbos            -- Maximum bounded object set level --
                  -- Bounding level of HyTime bounded object set when document is a hub or subhub. --

    NUMBER        -- Constraint: depth of nested entities to include in BOS (0=no limit, 1=hub only) --
    0
boslevel          -- Bounded object set level --
                  -- Default BOS level used by data entities declared in hub document. --

    NUMBER        -- Constraint: depth of nested entities to include
                  in BOS (0=no limit, 1=this entity only) --

    #IMPLIED      -- Default: No HyTime BOS --

<
]]><!-- bos -->
```

6.5.2 HyTime BOS control data attributes

The attribute form **HyTime BOS control data attributes** (*bosdatt*) consists of common data attributes used to control entities’ inclusion in the HyTime bounded object set. The HyTime common data attributes can be used with any data content notation.

The attribute **BOS level** (*boslevel*) specifies the depth of entity reference relative to this entity to include in the bounded object set [see **6.2.4 Bounded object set (BOS)**]. If not specified, the BOS level specified for the HyTime document element is used. The boslevel value is subordinate to the maximum BOS defined for the hub or subhub document. The boslevel value cannot cause entities to be included in the BOS if doing so would exceed the maximum BOS level.

The attribute **include in BOS** (*inbos*) indicates whether the entity must be unconditionally included in the BOS (“inbos”) or is to be unconditionally excluded from the BOS (“notinbos”). If no value is specified, the entity is included when allowed by the governing BOS level. When “inbos” is specified, the entity will be included in the BOS unconditionally. Entities declared by an entity so included are included in the BOS only if they do not exceed the maximum BOS level or if they are also declared with a value of “inbos”.

NOTE 100 Entities “unconditionally” included or excluded are still subject to inclusion or exclusion by bosspecs, which are always the final arbiters of inclusion or exclusion in a HyTime BOS.

Some objects are considered by hyperdocument authors to be so critical for proper operation of the hyperdocument that they must be integrated into the effective BOS before the hyperdocument can be considered complete enough for any use to begin. Such critical objects have “foreground priority” in the HyTime BOS. The remaining objects of the HyTime BOS have “background priority”. The **bounded object set priority** (*bosprty*) attribute sets the default priority for itself and for all of the entities in the entity subtree of which it is the root.

NOTE 101 For example, it may be necessary to comply with activity policies appearing in objects that are external to the hub document; such objects should be specified as having foreground priority.

NOTE 102 Applications may choose to assemble the HyTime BOS or not. Similarly, if they do assemble the HyTime BOS, the objects can be assembled in any order. The concept of BOS priority provides a way to suggest a performance optimization for applications that, due to time constraints, must assemble the “foreground priority” objects (objects that are essential to making a minimally adequate presentation of the hyperdocument) first. Once the foreground priority objects of the HyTime BOS have been incorporated into the effective BOS, access to the hyperdocument could begin, while the remaining “background priority” objects are still being incorporated.

The hub document always has, in effect, foreground priority in the HyTime BOS. For all objects in an entity tree, the default priority is specified by the “bosprty” common notation data attribute of the root entity of that tree. The bosprty attributes of child entities, if present, specify the default priority for themselves and their children.

NOTE 103 For any given set of objects in an entity tree, the default priority specification may be overridden by one or more bosspec elements referenced by the bosspec attribute of HyDoc (see **6.5 HyTime Bounded object set**).

An entity declared by a hub document whose **subhub** attribute’s value is “subhub” is not subject to the normal entity tree discovery process from the perspective of the hub document. Instead, it is regarded as an independent hub document with its own normal entity tree discovery process, whose HyTime BOS is added to the hub document’s tentative HyTime BOS. The subhub’s maxbos, boslevel, inbos, and bosspec specifications are all taken into account before the resulting subhub HyTime BOS is added to the hub document’s HyTime BOS. The hub document specifies that an entity should be considered a subhub by specifying a value of “subhub” for the subhub common notation data attribute. The maxbos and boslevel of the parent hub document are ignored by the subhub, which defines its own, local maxbos and default boslevel.

NOTE 104 A hyperdocument author may wish to include another hyperdocument in the new hyperdocument without having to worry about whether or not his own hyperdocument’s boslevel (etc.) will inadvertently render that included hyperdocument’s BOS incomplete, without requiring the author to duplicate and edit that hyperdocument’s (perhaps very complex) BOS rules in the new hub document. The subhub facility satisfies this requirement.

NOTE 105 Like all other objects reported by the entity tree discovery process, objects added to the HyTime BOS by virtue of their membership in the HyTime BOS of a subhub document can be removed from the HyTime BOS by bosspec elements referenced by the bosspec attribute of HyDoc.

NOTE 106 Subhubs can declare subhubs.

```
<!-- HyTime BOS Control Data Attributes -->
<![ % bos; [
<!-- attlist #NOTATION
-- bosdatt --          -- HyTime BOS control data attributes --
-- Clause:6.5.2 --

#ALL
boslevel              -- BOS level --
                      -- Bounded object set level for the entity --

NUMBER               -- Constraint: depth of nested entities to include in BOS (0=no limit, 1=this entity only) --
#IMPLIED             -- Default: value of boslevel attribute of HyDoc element. --

inbos                 -- Include in BOS --
                      -- Unconditional include in, or exclude from, BOS --
```

```

(inbos | notinbos)
#IMPLIED          -- Default: inclusion controlled by BOS level --

bosprty           -- Bounded object set priority --
                  -- Default BOS priority of objects in entity tree rooted at this entity. --

(foregrnd | backgrnd)
foregrnd
subhub            -- Is entity a subhub? --
(subhub | nosubhub)
nosubhub
>
]]><! -- bos -->

```

6.5.3 Bounded object set exception specification

The **bounded object set exception specification** (*bosspec*) attribute form specifies bosspec elements that define adjustments to be made to the results of all normal entity tree discovery and subhub processing.

```

<![% bosspec; [
<![attlist
-- bosspecat --      -- BOS except specification attributes --
                    -- Clause: 6.5.3 --

(HyDoc)
bosspec            -- Bounded object set exception specification --
                  -- Adjustments to be made to the bounded object set --

IDREFS            -- Reference --
                  -- Reftype: bosspec+ --
                  -- Constraint: must be internal reference --

#IMPLIED          -- Default: no BOS exception specification --
>
]]><! -- bosspec -->

```

The element form **bounded object set exception specification** (*bosspec*) identifies an entity in the BOS rooted at the hub document that refers to the bosspec in order to control its inclusion in the BOS, its “priority”, or both. Bosspec elements are resources used by reference via the bosspec attribute of the HyDoc element form. A bosspec element must occur in the hub (or subhub) document that refers to it.

A bosspec element contains one or more **BOS paths**. A BOS path consists of a list of entity names, of which the first must be declared in the hub (or subhub) document that contains the bosspec. The next entity name is the name of an entity declared within that entity, etc. The last entity named in a BOS path is the entity being addressed. When there is more than one path in a bosspec, each path is enclosed in parentheses. Entity names that do not conform to the concrete syntax of the hub document may be specified as literals if necessary.

The boslevel, inbos, and bosprty attributes are as defined for the common data attributes of the same names (see **6.5.2 HyTime BOS control data attributes**).

The hub document’s bosspecs are applied after all subhub and non-subhub entity tree discovery has been completed. The final adjustments to the HyTime BOS are made by the hub document’s bosspecs, and of those adjustments, the ones whose inbos attribute’s value is “notinbos” are applied last of all, so they can exclude any objects from the HyTime BOS, regardless of how they came to be tentatively included in it.

NOTE 107 Conceptually, every subhub’s BOS is established *before* the BOS of the hub or subhub with respect to which it is a subhub. Therefore, conceptually, the sub-most subhub’s BOS is established first, and the hub’s BOS established last, so that the hub’s bosspecs are the final arbiters. Whatever is contributed to the hub’s BOS (or any subhub’s BOS) by the hub’s (or subhub’s) own subhubs can be discarded selectively by the hub’s (or subhub’s) own bosspecs. Each hub or subhub applies its own bosspecs as part of the establishment of its own BOS, and these can be used to exceptionally include or exclude any entity included in or excluded from the BOS established by combining the BOSs of all of its own subhubs, together with the BOS established by its own entity declarations, boslevel, maxbos, etc. The bos control attributes do not apply to the contributions to the BOS that are the BOSs of subhubs. A subhub’s BOS is incorporated wholesale, and only a bosspec can override it.

If the same storage object is declared as an entity more than once in the BOS, excluding one of its declarations with a bosspec will not automatically exclude its other declarations; complete exclusion of the storage object requires exclusion of all the entities for which it is the replacement data.

NOTE 108 In other words, there is no SGML- or HyTime-defined mechanism for determining that two different entities are in fact the same storage object (although processing applications are free to define their own algorithms for doing so).

```

<!-- Bounded Object Set Exception Specification -->
<![ % bosspec; [
<!-- element
    bosspec          -- Bounded object set exception specification --
                    -- Clause: 6.5 --
                    -- Used to affect the HyTime BOS by overriding the inclusion or exclusion and priority of the entities
                    -- identified by the BOS path or paths given as content. --

    - O
    (#PCDATA)        -- Lextype: ((ENTITY, (cname|literal)*|
                    -- (GRPO, ENTITY, (cname|literal)*, GRPC) +)
                    --
                    -- Constraint: If parentheses are used, each parenthesized list is a separate BOS path. --
                    -- Constraint: Each word or literal in a BOS path is the name of an entity declared in the entity identified by
                    -- the previous word, literal, or entity name. --

-- Attributes [base]: bosspec --
-- CommonAttributes [GenArc]: daf, dvlatt, etfullnm, id, irefmodl, ireftype, lexttype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refctl, refloc, reftype, rflocspn --
-- Referrers [base]: HyDoc:bosspec --
>
<!-- attlist
    bosspec          -- Bounded object set exception specification --
                    -- Clause: 6.5 --

    boslevel         -- BOS level --
                    -- The BOS level from the last entity named in each specified BOS path to be affected by this bosspec. --

    NUMBER
    1                -- Constraint: depth of nested entities to include in BOS (0=no limit, 1=last entity only) --

    inbos            -- Include in BOS --
                    -- Unconditionally include or exclude objects declared by the last entity named in each BOS path, to the BOS
                    -- level specified by this bosspec's boslevel attribute. --

    (inbos|notinbos)

    #IMPLIED         -- Default: BOS unaffected --

    bosprty          -- Bounded object set priority --
                    -- Unconditionally specify the BOS priority of objects declared by the last entity named in each BOS path, to
                    -- the BOS level specified by this bosspec's boslevel attribute. --
                    -- Note: The semantic of the bosprty attribute is not affected by the value of the inbos attribute (that is,
                    -- whether it is explicitly "inbos" or the value is implied). --

    (foregrnd|backgrnd)
    #IMPLIED         -- Default: no priority change --
>
]] ><!-- bosspec -->

```

6.6 HyTime architectural bridging forms

The element form **HyTime architectural bridging element** (*HyBrid*) bridges between HyTime and non-HyTime elements. No HyTime processing is defined for the HyBrid form other than that provided by the HyTime common attributes. The HyBrid form must be used for any element type that uses any of the HyTime common attributes but does not have a more specific HyTime element form. When architectural markup minimization is in effect, the HyBrid form is automatically used for any client element that is considered architectural (because it has a unique identifier) but that does not have a more specific HyTime form (see **A.3.6.2 Architectural markup minimization**).

```
<!-- HyTime Architectural Bridging Element -->
<! element
  HyBrid          -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

  - O
  (% HyCFC;)*
-- Attributes [base]: HyBrid --
-- CommonAttributes [GenArc]: dafe, dvlatt, etfullnm, id, irefmod1, ireftype, lexttype, opacity --
-- CommonAttributes [base]: activity, conloc, dtxtatt, valueref --
-- CommonAttributes [locs]: refct1, refloc, reftype, rflocspn --
>
<! attlist
  HyBrid          -- HyTime architectural bridging element --
                  -- Clause: 6.6 --

  GenArc  NAME  #FIXED  GABrid
>
```

The notation form **HyTime architectural bridging notation** (*HyBridN*) bridges between HyTime and non-HyTime data entities. No HyTime processing is defined for the HyBrid form other than that provided by the HyTime common data attributes. The HyBridN form must be used for any data entity that uses any of the HyTime common data attributes but does not have a more specific HyTime notation form. When architectural markup minimization is in effect, the HyBridN form is automatically used for any client data entity that does not have a more specific HyTime form (see **A.3.6.2 Architectural markup minimization**).

```
<!-- HyTime Architectural Bridging Notation -->
<! notation
  HyBridN          -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

  PUBLIC "ISO/IEC 10744:1997//NOTATION
    HyTime Architectural Bridging Notation//EN"
-- Attributes [base]: HyBridN --
-- CommonAttributes [GenArc]: altreps, included, superdcn --
-- CommonAttributes [base]: bosdatt --
-- CommonAttributes [locs]: egrvplan --
>
<! attlist #NOTATION
  HyBridN          -- HyTime architectural bridging notation --
                  -- Clause: 6.6 --

  GenArc  NAME  #FIXED  GABridN
>
```

6.7 Common attributes

A set of architecture control attributes for all architectures is available to HyTime applications (see **A.3.5 Architecture control attributes**), as are the common attributes of the General Architecture (see **A.5 General Architecture**), and, when the location address module is supported, the ID reference control attributes (**7.7 Reference control**). This clause defines additional attributes for the HyTime architecture.

NOTE 109 By default, the HyTime meta-DTD includes the declarations necessary to use the “id” attribute of the General Architecture. Use of other General Architecture facilities must be declared in the documents or architectural meta-DTDs that use them.

NOTE 110 Elements that use the common attributes must also conform to a defined HyTime architectural form. The HyBrid form can be used for those elements that would otherwise not be clients of a more specific HyTime element form.

6.7.1 Value Reference

The attribute form **value reference** (*valueref*) associates attributes, an element’s content, or an element itself, with referential attributes (or content) that can be used to refer to the effective value of the attribute, content, or element node. The referential attributes so named are referred to collectively as “value reference attributes”.

NOTE 111 The “effective value” of an attribute is the value that an application or architectural processor needs, which is not necessarily limited to the types of data or objects that can be expressed through SGML’s attribute value syntax. For example, an element type may represent an application-defined object class that has properties that are themselves structured. The “property-of” relationship can be indicated by making the property an attribute of the element type. The structure of the property can be represented by using additional elements to structure the value and using value reference to refer to those elements as the “real” value of the attribute (where the syntactic value of the attribute might be the reference itself).

NOTE 112 A “#ELEMENT” value reference allows one SGML element to act as a “placeholder” for another element held somewhere else (for example, as a subdocument). However, the element making the reference is still addressable. Addressing the element making the reference is equivalent, for the purposes of HyTime-specific processing, to addressing the effective element node.

The *valueref* attribute value is a list of name pairs. The first name of each pair may be an attribute name, the keyword “#CONTENT”, or the keyword “#ELEMENT”. It names the attribute, content (“#CONTENT”), or complete element (“#ELEMENT”) for which the value is to be addressed by the attribute or content named by the second keyword. If the same attribute (or #CONTENT) is used for both names, the attribute or content refers to its own semantic value and never takes its semantic value directly (e.g., the *exspec* attribute of the event element form). The keywords “#CONTENT” and “#ELEMENT” may only appear once as the first keyword of a pair. The keyword “#CONTENT” may only appear once as the second keyword of a pair.

When the first name in a pair is “#CONTENT”, the effective value of the content property of the element is the node list addressed by the attribute or content named by the second name in the pair.

NOTE 113 For example, to provide an attribute called “conloc” for referring to the content of an element, use a declaration of the form:

```
< ! ATTLIST Para
    HyTime NAME #FIXED "HyBrid"
    valueref CDATA #FIXED "#CONTENT conloc"
-- Locate the content of the para --
    conloc IDREF #IMPLIED
>
```

The *valueref* attribute indicates that the content of the element *Para* will be located by reference using the referential attribute “conloc” (the *conloc* attribute is referential because it has a declared value prescription of “IDREF”).

When the first name in a pair is “#ELEMENT”, the effective element node is the single node addressed by the attribute or content named by the second name in the pair. When the object addressed is a document or subdocument entity, the effective element node is the document element of the addressed document. It is an RHE to address more than one node as the effective element node.

The objects located as the effective value of attributes, content, or elements need not satisfy the syntactic constraints defined in the referring document’s document type declaration. However, they must satisfy any semantic constraints defined by the processing application. The mechanism for defining such constraints is outside the scope of the HyTime standard.

NOTE 114 Semantic constraints could include conformance to the lexical type of an attribute, the content model of the referring element, conformance to one or more architectural meta-content models, or other semantic constraints unrelated to SGML-defined constraints. Application designers can use the reference type control facilities of HyTime and the General Architecture to control the types and organization of elements referred to.

Value reference does not modify how an SGML document is parsed.

NOTE 115 Conceptually, the result of a value reference is maintained in the HyTime semantic grove, not the HyTime effective SGML document grove. This means in particular that a node addressed by value reference does not occur in any name spaces in the HyTime extended SGML document grove of the document from which the value reference is made if the node's origin is not already in that grove.

Thus, if an element in another document is made the content of an element by value reference, the referenced element can be addressed by name only by a inter-document address but can be addressed directly by tree location with respect to the referring element.

For the purposes of accessing attribute values and the content of elements via HyTime location addressing, the data addressed by value reference are always used even if there is a directly-specified value for the attribute or content. The only exception is property location addresses, which can, optionally, address the direct value of the attribute or element, rather than its referenced value.

NOTE 116 In other words, valueref serves to "redirect" location addresses from the direct value to the indirectly-addressed value. Addresses resolved by means other than a HyTime engine (such as a DSSSL engine) need not respect the redirection performed by value reference attributes.

When the location addressing module is supported, any supported means of addressing may be used for value reference. If location addressing is not supported, value reference attributes must be declared as IDREF, IDREFS, ENTITY, or ENTITIES and content cannot be used for value reference.

An element that uses value reference to refer to its content may allow syntactic content. There is no requirement that elements that provide an attribute for value reference of their effective content declare that attribute with a default value of #CONREF — any syntactic content will be ignored for HyTime processing when the value reference attribute is specified in any case (unless the content itself contains the reference).

NOTE 117 In the conloc example above, the Para element could have syntactic content, but HyTime location addresses that address the content of the Para element would address the objects addressed by the Para's conloc attribute, not the data in its syntactic content (except as provided by the proploc element form).

```
<!-- Value Reference Attribute Namer -->
<![ %valueref; [
<!-- attlist
-- valueref --      -- Value reference attribute namer --
-- Clause: 6.7.1 --

#ALL

valueref      -- Value reference attribute namer --
-- Associates attributes (or element's content) with referential attributes (or content) that can be used to refer
-- to the semantic value of the attribute or content. --
CDATA        -- Lextype: ((ATTORCON|"#ELEMENT"), ATTORCON) + --
-- Note: The first ATTORCON is the attribute or content for which the value is to be addressed, the second
-- ATTORCON is the attribute or content by which the value reference is made. --
-- Constraint: second attribute in each pair must be a referential attribute or content defined as referential by
-- the refloc facility. --
-- Constraint: "#ELEMENT" and "#CONTENT" may only occur once as first keyword of pair. "#CONTENT"
-- may only occur once as second keyword of pair. --
#IMPLIED     -- Constant --
>
]]><!-- valueref -->
```

The following form is included for backward compatibility.

```
<!-- Content Location -->
<![ %conloc; [
<!-- attlist
-- conloc --      -- Content Location --
-- Clause: 6.7.1 --

#ALL
HyBase NAME     #FIXED HyBrid
valueref CDATA  #FIXED "#CONTENT conloc"
```