Davenport Advisory Standard for Hypermedia (DASH), Module I: Standard Open Formal Architecture for Browsable Hypermedia Documents (SOFABED)

Editors: Steven R. and Victoria Taussig Newcomb*

DASH/SOFABED 0.9.6, 19 February 1993

Contents

*This (final?) draft incorporates editing instructions given at the DASH/SOFABED Workshop held December 13-15, 1992 at O'Reilly & Associates, Sebastopol, California. For more information, contact Dale Dougherty, O'Reilly & Associates, Inc., 103A Morris Street, Sebastopol, California 95472. Tel. +1 800 998 9938. Fax +1 707 829 0104. UUCP: uunet!ora!dale. Internet: dale@ora.com. This standard conforms to the "HyTime" Hypermedia/Time-based Structuring Language Standard (ISO/IEC 10744:1992).

1 Introduction to the DASH and the SOFABED.

This **Standard Open Formal Architecture for Browsable Electronic Documents (SOFABED)** forms a modular, independently useful portion of the broader **Davenport Advisory Standard for Hypermedia (DASH)**, which will comprise several other modules in addition to the SOFABED module.

1.1 The DASH

The DASH as a whole amounts to an application-specific set of elaborations and constraints on the "HyTime" Hypermedia / Time-based Structuring Language (ISO/IEC 10744:1992), which in turn is based on the "SGML" Standard Generalized Markup Language (ISO/IEC 8879:1986). The field of application of the DASH is the representation of document "source code" from which both printed and on-line technical documentation can be automatically derived and formatted. The scope of the DASH as a whole is not yet completely defined. The general intent of the Davenport Group is to standardize whatever constructs that Davenport participants[1] have sufficient interest in standardizing to justify the effort of doing so, and which are congruent with its strategic objective. The Davenport Group's strategic objective is to define an SGML/HyTime-based model and language for the representation of online and printed technical documentation source of the kind usually supplied by Unix system and other open systems vendors to their customers.

The DASH will allow Open Systems vendors to "bundle" (to sell along with their systems) technical documents provided by a variety of publishers, and to give their customers access to all this information via user interfaces that they may define and implement differently and independently of one another. The DASH will allow documents of dissimilar SGML document types[2] to be delivered as a single compound document with whatever aggregated indexes, tables of contents, cross-references, and webs of hyperlinks the systems integrators may wish to generate and/or compile, without any special assistance from the originators and/or publishers. Document originators and publishers will be able to create and maintain a single version of each document without regard to the modes and styles of its ultimate delivery to users, since the DASH makes it possible for the

[1]Participation in the standardization activities of the Davenport Group is open to the public.

[2]SGML is the International Standard Generalized Markup Language defined by ISO/IEC 8879:1986. An SGML document type definition (DTD) describes the structure of a class of documents, all of which have the same allowable structural relationships between their various logical components ("elements"), and all of which share the same lists of properties ("attributes") for each type of element.

same original document to be formatted for online delivery as hypermedia as well as for printing on paper.

Future DASH modules will standardize:

- Representation of the policies of document owners regarding the use of the information contained in their documents. This next planned module of the DASH is called "Conditions of Use for Commercial Hypermedia (COUCH)." Among other things, the COUCH will facilitate royalty accounting, and it will provide a standard protocol for notifying the authors of referencing documents when the referenced documents have been changed. A basis for standardizing the representation of these kinds of information already exists in HyTime's "activity tracking" facility.[3]

- Representation of schedules of events in time and space, using HyTime "Finite Coordinate Spaces (FCSs)." FCS "schedules" can be used to represent time-based animations, simulations, and multimedia presentations in such a way that documents that incorporate multimedia information objects can be created and maintained in a single source form which can be formatted for use on a wide variety of systems. HyTime's scheduling facilities allow explicit distinctions to be made between, on the one hand, information that is essential to the meaning of the information that constitutes a multimedia presentation, and, on the other hand, formatting information which is purely renditional and which can safely be ignored by applications that have no need to render a multimedia presentation in order to make use of the information it contains. FCSs can also be used to represent arbitrarily complex tables in an elegant and standard fashion, such that HyTime information processing software will allow tabular data to be queried without first being formatted for presentation.

Like the HyTime standard on which it is based, the Davenport Advisory Standard for Hypermedia (DASH) is intended to be used as an infrastructure for platform-independent information interchange; it is *not* optimized for encoding the internal representation of information on which specific hypertext application programs act.[4] Developers of document processing applications will provide facilities for importing and exporting documents in such a way as to conform to Davenport constructs, but the manner in which a document is represented within and/or manipulated by an application is defined neither by HyTime nor by the DASH.

[3]Earlier drafts of this SOFABED Standard described one use for HyTime "activity tracking" elements: representing the owner's requirement (or suggestion) that when an owned information object (an element) is identified as an anchor (linkend) of a link created and owned by someone else, that a certain owner-provided visual, auditory, or other cue be used to indicate to the link's user that traversal to the owned information object is possible. Although this issue is relevant to SOFABED's mission of standardizing a set of constructs for hypertext navigation, this issue has been referred to the COUCH committee for consideration along with all other issues related to "activity tracking" for reasons of practicality and consistency.

[4]Nothing in HyTime inherently prevents real-time execution of HyTime documents directly from the HyTime representation. However, as a practical matter, this is difficult using today's information processing systems. Against the day that hardware price/performance begins to meet the demands of rendering HyTime documents directly from their native SGML representation, HyTime contains certain features, such as "bit-skipping," that are intended to help reduce those demands.

The DASH consists of a set of templates, called "SGML Architectural Forms," to be used by document architects as they create the SGML document type definitions which establish the syntax and structure of their documents. Because every element type or attribute defined in accordance with one of these forms takes on the semantics of the form, it is possible to build standard "Davenport Engines" — reusable generic software modules that perform the processing that is associated with the semantics of each of the Davenport architectural forms.

Some of the DASH architectural forms are based on, and are entirely consistent with, the HyTime International Standard Hypermedia/Time-based Structuring Language. None is in conflict with HyTime.

When performing the semantic processing required by the DASH on elements which conform to HyTime architectural forms, the behavior of a Davenport Engine will be indistinguishable from the behavior of a HyTime Engine acting on the same data, except:

1. A Davenport Engine will recognize Davenport architectural forms that do not inherit the characteristics of any HyTime form; and

2. A Davenport Engine will apply additional constraints, not required by HyTime, in accordance with the prescriptions of particular Davenport architectural forms that do inherit characteristics of HyTime forms.

Therefore, a Davenport Engine need not duplicate the functionality of a HyTime Engine, but could instead consist of a set of validating and processing routines added as a module to a system that already contains the appropriate subset of the functionalities of a complete HyTime Engine.[5] The exact behavior of a Davenport Engine is not defined by the DASH; the design and implementation of such engines, if any, is an arena for competition in the marketplace.[6]

1.2 The SOFABED

By contrast to the scope of the DASH, which is as yet not completely defined, the scope of the SOFABED *is* well defined. The SOFABED standardizes only the representation of certain kinds of "navigational" information, *viz.*: tables of contents, indexes, and glossaries.

The SOFABED is being developed partly in response to the widespread sentiment among software and systems vendors, and those who supply and maintain their documentation, that standardizing the representation of hypertext links – hyperlinks – will benefit everyone involved. The reasons for this broad consensus include:

- the increased usefulness of information that can be accessed in the context of other documents increases the value of that information in the marketplace, and

[5] Some of the features of HyTime, such as the ability to schedule events in virtual or real time and/or space, are not used in the SOFABED, but may be used in future DASH modules. This presents no problem, because HyTime's modular design allows applications and documents to support the various features of HyTime selectively. However, in the likely event that online technical documentation will eventually include such things as schedulable hypermedia objects, the decision to become HyTime-compatible at this early stage offers the advantage that the structure of today's text and graphic information will be 100% compatible with that of tomorrow's multimedia documents.

[6] Neither does the HyTime standard specify the behavior, design, or implementation of a HyTime engine.

- the increased usefulness of information that contains pointers to broad base of other documents improves its own marketability and its owner's revenues, even when other information owners may receive additional revenue when such links are traversed by users.

2 References and Resources

1. ISO/IEC[7] 8879:1986, *Standard Generalized Markup Language* (SGML). This is the international standard metalanguage on which both HyTime and, in turn, the DASH are based.

2. ISO/IEC 9069:1992, *SGML Document Interchange Format* (SDIF). This standard shows how documents represented in SGML, including HyTime and DASH documents, can be interchanged between similar and dissimilar systems in a fashion congruent with the Open Systems Interconnect (OSI) seven-layer model, employing the standard Abstract Syntax Notation 1 ("ASN.1").

3. ISO/IEC 9070:1992, *Registration Procedures for Public Text Owner Identifiers.* This standard provides a procedure whereby SGML text such as DTDs can become publicly available by reference to an assigned unique internationally standardized number.

4. ISO/IEC 10744:1992, *Hypermedia / Time-based Structuring Language* (HyTime). Itself based on SGML, HyTime is the standard foundation on which the DASH is built.

3 How to Read the DASH

The DASH is expressed using a formalism that was first used to represent the HyTime standard. Rather than containing the `ELEMENT` and `ATTLIST` declarations normally found in SGML document type definitions (DTDs), the DASH sets forth templates, or "SGML architectural forms," which are intended solely for human readers (they are not meaningful to SGML parsers). A collection of such forms may be said to constitute a "meta-DTD." SGML architectural forms are, in essence, rules for constructing element types intended to convey certain standardized meanings. Architectural forms may include requirements affecting only attribute lists, or both attribute list requirements and content model requirements. The grammar of SGML architectural forms resembles the grammar usually used to express SGML document type definitions; the many similarities make it convenient for document architects to use the architectural forms as templates as they create specific Davenport-compliant DTDs. (Such DTDs will normally include both Davenport and non-Davenport element type declarations [i.e., some elements that do and some that do not conform to Davenport architectural forms]; the content model and attribute list of each of the Davenport-conforming element types may or may not be more constrained than the corresponding

[7] "ISO" stands for *International Organization for Standardization.* "IEC" stands for *International Electrotechnical Commission.* The two organizations cooperate to develop and promulgate international electronic standards through their "Joint Technical Committee 1" (JTC1). JTC1 has about thirty subcommittees, including Sub Committee 18 (SC18), which has general responsibility for electronic publishing standards. All of the ISO/IEC standards in this list were developed under the auspices of JTC1/SC18.

Davenport architectural form, and it may or may not contain additional attributes not required by the corresponding Davenport architectural form.)

3.1 Editorial Conventions

One Section of each module of the DASH consists of normative definitions and descriptions of the Davenport Architectural Forms (the SGML templates) of that module (Section 4 in this SOFABED module). Each architectural form (or small set of related architectural forms) is defined using the formalism developed to express the HyTime meta-DTD; this is preceded by a description of the form in English.[8]

3.2 Meta-DTD Conventions

As in the HyTime standard, architectural forms are specified using both

1. element-type forms – templates that provide guidelines to SGML DTD authors as they create element type definitions, and which may dictate certain requirements regarding both the content models and attribute lists of such element type definitions; and

2. attribute list type forms – lists of one or more attributes that may be used in a variety of element type definitions.

In a meta-DTD, the keywords `element` and `attlist` are written in lower case in order to emphasize the fact that this is a meta-DTD rather than a true SGML DTD.

3.2.1 Names of Element-type Architectural Forms

The formal name of each element-type Davenport Architectural Form, which will be used as the value of a required Davenport attribute (`Davnport`), consists of two or three fields, with a period (.) used as the field separator. Each form will be introduced using its plain English name followed by its formal name in parentheses; it may subsequently be referred to in explanatory text either by the second field of its formal name or by its plain English equivalent.

1. The first field is always the same: "dav". All Davenport architectural forms can be readily identified as such by this prefix.

2. The second field uniquely identifies the type of the Davenport architectural form.

3. The third (last) field is the name of the HyTime architectural form, if any, to which the Davenport architectural form conforms. If it does not conform to any HyTime architectural form, the third field is empty, and there is no field separator after the second field.

[8] For a thorough understanding of each Davenport architectural form, and especially of those that are based on HyTime architectural forms, it is imperative to consult the HyTime standard. The DASH does not and is not meant to recapitulate the HyTime standard.

The length of the names used for the Davenport element type architectural forms often exceeds eight characters, in order to be as clear as possible as to the nature of the form. Because it will be the actual, declared generic identifiers of element types built on these forms that will be parsed, the length of these form names is irrelevant as far as SGML parsing is concerned. Since the attributes defined for these forms will, unless explicitly aliased, keep the names given them in the DASH, their names are no longer than eight characters in order to guarantee universal parsability according to SGML's reference concrete syntax.[9]

3.2.2 Element-type Architectural Form Attribute Lists

Attributes are contributed by HyTime architectural forms to Davenport architectural forms as required by HyTime; an effort has been made to identify those attributes in the Davenport architectural forms that they inherit from their corresponding HyTime architectural forms.

Constraints on attribute values and data types imposed by the DASH are indicated by conventional comments[10] and/or by added HyTime `lextype` and `reftype` attributes.

HyTime "common attributes"[11] are included in the formal specification of a particular architectural form only when they interact with other architectural forms. For example, an `id` attribute will be included for a form only when some architectural form includes an attribute that references it. As with HyTime, DTD designers are always free to add application-specific attributes and HyTime common attributes to Davenport-compliant element declarations.

3.2.3 Element-type Architectural Form Content Models

Content models for Davenport architectural forms are shown as ordinary SGML content model declarations. The content models may be further defined by associated conventional comments, which may indicate additional lexical and/or semantic constraints on the content of any actual conforming element type. Any actual element type's content model may be *more* constrained than the model as described and constrained by the architectural form, but it may not be *less* constrained.[12]

[9] In the SGML "reference concrete syntax," `NAMELEN` is 8, which means that names are limited in length to eight characters. This limitation is the reason why the data type of the `Davnport` attribute is `CDATA` rather than `NAME`; if governed by the reference concrete syntax, the value "`dav.sign.ilink`," for example, would be an illegal `NAME` value because it is longer than eight characters.

[10] *Conventional comments* are SGML comments in meta-DTDs that have the force of law — i.e., they impose conformance requirements. Complete familiarity with the HyTime will be a great help in understanding the usage of conventional comments in this Standard.

[11] HyTime "common" attributes may be added to any element type that also has a `HyTime` attribute. These "common attributes" are recognized and processed appropriately by conforming applications even if the value of the `HyTime` attribute is "`nHyTime`" (meaning that the element type conforms to no HyTime element-type architectural form). See the HyTime standard for details.

[12] Some examples:

- When an architectural form's content model contains a user-definable parameter entity reference, a conforming element type may be considered "wide open" with respect to the element types, etc., that it may contain, within the constraints (if any) on that parameter entity and on the content model itself.

3.2.4 Attribute-list-type Architectural Forms

Davenport attribute-list-type architectural forms are all very similar in their definition and use to HyTime common attributes. They will be recognized by a Davenport engine on any element in which a HyTime common attribute would be recognized,[13] and if they are not found in any element in which they could be recognized, a Davenport engine will behave as though they exist and have the default value prescribed in the Davenport meta-DTD.

3.2.5 HyTime Architectural Forms

In addition to the architectural forms defined in the DASH, DASH-compliant documents may use the HyTime architectural forms that are implied by the HyTime support declarations given in each module of the DASH (these are given in Section 5 of this SOFABED module).

4 SOFABED Architectural Forms

4.1 Davenport Semantic Roles

Many different uses are possible for any declared element type. The same can also be said of any HyTime (or Davenport) architectural form. Each different purpose for which a form may be used implies different semantics from other possible purposes, but the syntax of the form, and usually some underlying semantic, is common to all possible uses of any given element type or architectural form.

The DASH[14] defines a special common attribute form, **Davenport semantic role** (`davrole`),[15] that may be used in the attribute lists of elements conforming to Davenport and HyTime forms (including those conforming to the `HyBrid` and `nHyTime` forms), and whose value indicates the purpose(s) to which the element is intended in the context of Davenport applications. The value

---

- Similarly, the content model `ANY` in an architectural form means that a conforming element type may contain anything. However, in either of the foregoing cases, an actual DTD may constrain the content model of a given element type to a particular element structure and/or data type, since it would then be more (and not less) constrained than that prescribed by the architectural form. It is very often appropriate to constrain the content model of a given element type to a greater extent than the architectural form's is constrained.
- The content model `EMPTY` in an architectural form would mean that a conforming element type may not contain anything; it is already fully constrained, and all elements conforming to that architectural form must have `EMPTY` as their content model.

[13]That is, a Davenport common attribute will be recognized and processed according to DASH specifications if the element on which it appears both has a HyTime attribute, and is contained within an unbroken hierarchy of elements, all the way back to the document element (root element), which all have HyTime attributes.

[14]This is a DASH rather than a SOFABED form, as it will be used in forms that fall outside the scope of the SOFABED

[15]The name of the attribute itself is "`davrole`," but, in order to be consistent with the convention established for the HyTime standard that the names of common attribute-list-type forms have the prefix "`all-`," the name of the architectural form is "`all-dav`." The string "`all-dav`" in this architectural form's definition is not for use in any actual DTD, and it has no significance beyond that described in this footnote.

of the `davrole` attribute is a list consisting of one or more Davenport "semantic key names" (SKNs) and/or one or more application-defined SKNs. A mechanism for defining and aliasing groups of SKNs is provided by the `dav.srdef` element-type form (see Section 4.2). Some SKNs for `davrole` are defined by this standard (see Section 4.2.1) and must be recognized by all SOFABED-compliant applications. The generic identifiers of the element types used to define the SKNs used in a `davrole` attribute's value must be given in the value of an accompanying **Davenport semantic role definition generic identifiers** (`dsrdgis`) attribute. If no `dsrdgis` attribute is present, the generic identifier `DavDSRD` (Davenport davrole semantic role definition – see Section 4.2.1) will be assumed, and the SKNs defined by this standard, together with those defined by any `DavDSRD` elements present in the document, will be allowed in the value of the associated `davrole` attribute.

By standardizing this means of expressing the purpose(s) to which an element can be put, the DASH greatly enhances the interoperability of documents that are governed by differing DTDs, while avoiding the necessity of defining architectural forms that differ in name only.[16]

The values of `davrole` attributes are intended to serve as suggestions and/or guideposts to applications that may not have a good grasp of the governing DTD or the design philosophies of the document. Some applications may take the position that no role that has not been explicity suggested by `davrole` is permissible, while other applications may use `davrole` in some other way, or they may ignore it altogether. In cases where an application has intimate knowledge of the DTD and its guiding philosophies, `davrole` may be safely ignored (except, of course, by the application designer). Therefore, the values given to `davrole` attributes should be designed to provide the document with the information about the originator's intentions that will be most useful to an application that receives the document as if it were an artifact from another planet.

```
                <!-- Davenport Semantic Role -->
<!-- Davenport common attribute that helps identify the purpose of
     elements, thus enhancing the interoperability of documents
     that may conform to different Davenport-compliant DTDs. -->
<!attlist all-dav
    davrole     -- Semantic role of this element. Use one or more
                   Davenport SKNs and/or application-defined SKNs.
                   Use dav.srdef elements to define and/or alias them. --
                CDATA    #IMPLIED "" -- Default: no Davenport-defined
                                        or user-defined role. --
    dsrdgis     -- GIs of dav.srdef-form elements which define
                   the SKNs used in the davrole attribute appearing in
                   the same element. --
                CDATA    #IMPLIED "DavDSRD" -- Default: Davenport
                   built-in definitions plus those given in all
```

[16]In effect, the `davrole` attribute identifies some superclass(es) from which a particular element instance (or, if its value is `#FIXED` in the DTD, all elements of a given type) inherits some characteristics. In that sense, it is similar to the `Davnport` and `HyTime` attributes, except that, unlike the values of `Davnport` and `HyTime` attributes, which may trigger certain syntactic expectations in a DASH application regarding the element in which it appears, the value of a `davrole` attribute creates semantic expectations only.

```
                dav.srdef-form DavDSRD-type elements. -- >
```

4.2 Semantic Role Definition

Davenport semantic roles are defined by **semantic role definition** (`dav.srdef`) elements. These elements contain a description of the role being defined; the attribute **semantic key name** (`skn`) assigns a name to the role.

The attribute `roletype` allows the role being defined to be reserved for use in either of two Davenport-specific contexts ("`davrole`" for use in the value of a `davrole` attribute, or "`anchrole`" for use in the value of an `anchrole` attribute of a `dav.sign.ilink`-form element), or in any other context requiring the specification of a semantic role ("`other`").

Useful semantic roles which must be recognized by all Davenport engines are defined in Sections 4.2.1 and 4.4.1.[17]

It is a Reportable Davenport Error (RDE)[18] if an attempt is made to redefine an SKN that is already defined by this standard using a `DavDSRD`-type element or a `DavASRD`-type element. It is an RDE if any SKN is defined more than twice by any pair of `dav.srdef`-form elements that share the same generic identifier, or by any pair of elements whose generic identifiers are `DavDSRD` and `DavASRD` respectively.[19]

Neither the placement nor the order of `dav.srdef`-form elements within a document instance is significant.[20]

The **alias for** (`aliasfor`) attribute allows the compounding of semantic role definitions, thus allowing the SKN(s) given in the `skn` attribute to serve as an alias for all of the SKN(s) that appear in its value. The number of SKNs appearing in the value of the `aliasfor` attribute must be an integer multiple of the number of SKNs appearing in the `skn` attribute. In cases where multiple SKNs appear in the value of the `skn` attribute, the mapping of the SKNs given in the `aliasfor` attributes onto their respective aliases appearing in the value of the `skn` attribute is done by using the sequence of SKNs that appears as the value of `skn` as a template, which is applied in repeated sequence to the list of SKNs given as the value of the `aliasfor` attribute.[21]

[17] The definitions given in Sections 4.2.1 and 4.4.1 can also be used as models for defining additional semantic roles in document instances, using `dav.srdef`-type elements in the document instance. However, those definitions do not demonstrate the full power of the `dav.srdef` architectural form. For example, although the `DavDSRD`-type elements in Sections 4.2.1 and 4.4.1 do not do so, it is possible to use the names of more than one SKN (that are singly defined in other `dav.srdef`-type elements or by this standard) as the value of the `aliasfor` attribute of any `dav.srdef`-type element; this allows a single user-defined SKN to become an alias for several others. For another example, the possibilities inherent in the ability to define and use a variety of classes of SKNs using a variety of `dav.srdef`-form element types, together with their `alia4gis` attributes, are not exploited in the definitions given in Sections 4.2.1 and 4.4.1.

[18] An RDE is handled similarly and corresponds to a Reportable HyTime Error (RHE); see the HyTime standard for details.

[19] For an explanation of this last rule, see the discussion of the `aliasfor` and `alia4gis` attributes in the second and third paragraphs following this one.

[20] There is no requirement, for example, that forward references to as-yet-undefined SKNs be avoided.

[21] For example, if the value of the `skn` attribute is `"a b c"` and the value of the `aliasfor` attribute is `"d e f g h i"`, then `a` can be used as an alias for `d` and `g`, `b` can be used as an alias for `e` and `h`, and `c` can be used as an alias for `f` and `i`.

The alia4gis attribute is used to specify the generic identifiers of the dav.srdef-form elements which define the SKNs that appear in the value of the aliasfor attribute. If the alia4gis is not present, the SKNs defined by all DavDSRD-type and DavASRD-type elements, plus all Davenport built-in SKNs (see Sections 4.2.1 and 4.4.1), may appear in the value of the aliasfor attribute.[22]

```
            <!-- Semantic Role Definition -->
<!element dav.srdef    -- Content describes the meaning or purpose of the
                          name(s) that appear as the value of this element's
                          skn attribute. --
                     - O   (#PCDATA) >
<!attlist dav.srdef
    Davnport    CDATA    #FIXED        dav.srdef
    skn         -- name(s) to be used in the value of a semantic role
                   attribute (davrole, anchrole, etc.), representing
                   the semantic role described in the content of this element,
                   and/or a combination of those of the SKNs specified by the
                   aliasfor attribute. --
                   Constraint: may not be the same as any other SKN defined
                   by an element of the same GI in the same document. --
                -- Constraint: must be valid SGML name(s) in the current
                   parsing context. --
                CDATA    #REQUIRED
    roletype    -- Is this skn meant to be used as a Davenport semantic
                   role ("davrole"), a HyTime anchor role ("anchrole"),
                   or some Davenport-unspecified role type ("other")? --
                (davrole|anchrole|other) other
    aliasfor    -- One or more SKNs for which the SKN(s) in the value of
                   the SKN attribute can serve as alias(es).  If the SKN
                   attribute's value is a single name, and the value of
                   aliasfor is more than one name, the multiple roles
                   represented in aliasfor are combined with the definition
                   in the content, if any, and the whole conglomerate
                   definition is invocable by the value of the skn attribute. --
                -- If this element's skn value contains more than one name, then
                   the aliasfor attribute must have an integer multiple (which
                   could be 1) of that number of names.  (See text for mapping
                   convention.) --
                CDATA    #IMPLIED      ""  -- default: none --
    alia4gis    -- One or more generic identifiers of elements used to
                   define the SKNs that appear in the value of the aliasfor
                   attribute. --
```

[22]This is the reason for the requirement that no SKN may be defined more than once by the collection of all DavDSRD and DavASRD elements, including the Davenport-inherent ones that appear in Sections 4.2.1 and 4.4.1.

```
CDATA    #IMPLIED      "DavDSRD DavASRD" -- Default: all SKNs
   defined by this standard and all SKNs defined in the
   document instance using DavDSRD-type and DavASRD-type elements
   may appear in the aliasfor attribute. -->
```

4.2.1 Built-in Semantic Roles (for davrole)

The element type DavDSRD is defined and instantiated in this Standard in order to provide built-in Davenport semantic roles for SOFABED applications. (See Section 4.2 for a discussion of the dav.srdef architectural form on which the DavDSRD element type is based.)

```
<!ELEMENT DavDSRD  -- Davenport-defined built-in semantic role
                      definition for use with davrole --
                   - O   (#PCDATA) >
<!ATTLIST DavDSRD
    Davnport    CDATA    #FIXED        dav.srdef
    skn         -- name to be used in the values of davrole attributes,
                   representing the semantic described in the content
                   of this element, and/or the combination of SKNs
                   specified by the aliasfor attribute. --
                CDATA    #REQUIRED
    roletype    (davrole|anchrole|other) #FIXED davrole
    aliasfor    CDATA    #IMPLIED      ""
    alia4gis    CDATA    #IMPLIED      "DavDSRD DavASRD" >
```

The following are definitions of Davenport semantic key names (SKNs) that can appear in the value of the davrole attribute and/or the values of the aliasfor attributes of dav.srdef-type elements. These SKNs must be recognized by SOFABED-compliant applications.[23] [24] An author of a Davenport document may add to the repertoire of SKNs used in a document by including elements conforming to the dav.srdef architectural form in that document.

```
<DavDSRD skn=toc>
Table of contents, or something intended to be used in conjunction
with a table of contents.  Possible uses include: an element
containing a set of dav.entry.ilinks to be used as a table of
contents; a dav.entry.ilink; a dav.sign.ilink; either linkend of a
dav.entry.ilink or dav.sign.ilink.
```

[23] That is, Davenport compliant documents can use these SKNs meaningfully without actually incorporating the DavDSRD elements set forth in this Section.

[24] Some SOFABED-compliant applications will have no use for certain SKNs. In such applications, "recognition" of such application-irrelevant SKNs will amount to no more than simply not complaining about the fact that an SKN defined by this standard was not defined in the document instance.

<DavDSRD skn=index>
Index, or something for an index. Possible uses include: an element containing a set of dav.entry.ilinks to be used as an index; a dav.entry.ilink; a dav.sign.ilink; either linkend of a dav.entry.ilink or dav.sign.ilink.

<DavDSRD skn=glossary>
A glossary, or something for a glossary. Possible uses include: an element containing a set of dav.entry.ilinks to be used as a glossary; a dav.entry.ilink; a dav.sign.ilink; either linkend of a dav.entry.ilink or dav.sign.ilink.

<DavDSRD skn=lof>
A list of figures, or something for a list of figures. Possible uses include: an element containing a set of dav.entry.ilinks to be used as a list of figures; a dav.entry.ilink; a dav.sign.ilink; either linkend of a dav.entry.ilink or dav.sign.ilink.

<DavDSRD skn=lot>
A list of tables, or something for a list of tables. Possible uses include: an element containing a set of dav.entry.ilinks to be used as a list of tables; a dav.entry.ilink; a dav.sign.ilink; either linkend of a dav.entry.ilink or dav.sign.ilink.

<DavDSRD skn=travCue>
A cue to user that traversal is possible. Use with any element that is intended to be used as an endterm of some linkend of some traversable link. (See the HyTime standard for details about endterms.)

<DavDSRD skn=latebptr>
A late-bound pointer. Characterizes a HyTime location pointer element as something that, while it can be converted at formatting time to another kind of pointer (such as a page number), it should remain, essentially, an unresolved pointer, whose pointingness is not concealed from the end user. Used only with HyTime location pointer elements.

<DavDSRD skn=erlybptr>
An early-bound pointer. Characterizes a HyTime location pointer element as something that should be resolved at formatting time into something that will not result in the end user being aware of its

```
existence, nor in the end user's having to decide whether to resolve
it or not.  Used only with HyTime location pointer elements.

<DavDSRD skn=topcntxt>
Topical context.  Used only to characterize a HyTime nameloc-type
location pointer element as being used to specify an aggregate "topic"
anchor of an entry link for the purpose of establishing the topical
context in which the lexically last-listed item in the nameloc should
be listed in an index or other navigational table.
```

4.3 Knowledge and Navigation: Concepts

A document's navigational information traditionally consists of

1. three kinds of tables:

   (a) **tables of contents**, in which navigational information is presented as a kind of map that summarizes the structure of the document;

   (b) **indexes**, in which navigational information is organized topically; and

   (c) **glossaries**, in which navigational information is sometimes given in order to allow users to find examples of the words being glossed; and

2. **cross-references** that may appear anywhere, including within the content of a document, and which have a multitude of purposes and forms.[25]

In all cases, the fundamental unit of navigational information can be represented and/or regarded as a hyperlink.[26] An index can be regarded as a collection of hyperlinks, not only between each index entry and some point(s) in a document, but also between each other. Certain relations between index entries are sometimes represented as index topic levels, while others are represented as *See* and *See Also* cross-references. Collectively, the way in which topics, subtopics, and sub-subtopics (together with various kinds of cross references) within an index are arrayed, can be regarded as a kind of knowledge base: it is a representation of a view of how the knowledge represented in the indexed document organizes itself, without necessarily any regard to the structure in which the same knowledge is presented in the main body of the document.

[25] This standard does not define an architectural form for cross-reference links, because no Davenport-specific semantics were identified for such a form. HyTime `clink`s and `ilink`s are sufficient in themselves for this purpose. Particular kinds of cross-references should be defined by DTD designers, and they should thereafter be identifiable by their generic identifiers (and possibly by the value of their `davrole` attributes, if any).

[26] The words *hyperlink* and *link* are used interchangeably in this Standard. While it may seem redundant, the word *hyperlink* is used to introduce the concept in order to distinguish it from the concept of *SGML link* as specified in an SGML Link Process Definition (LPD) – an entirely different concept with, perhaps regrettably, the same identifying nickname, *link*.

The usual way to create an index (in *troff*, *TeX*, or *Ventura*, for example) is to insert marked-up indexing commands or macros, key words, levels of key words, etc. into the text wherever there is something that should be indexed. The index is then created by some batch compilation process. Indexes made this way are typically somewhat self-inconsistent when they are first compiled, especially if more than one author created and/or maintained the document.[27]

The SOFABED provides a basis for creating and maintaining the navigational information in documents that is intended to address not only the intra-document consistency and redundancy problem, but also the more difficult problem of maintaining consistency and avoiding redundancies when whole libraries of documents are incorporated into larger libraries which can then be provided with a single master index.

4.3.1 Relationship of Sign Links and Entry Links.

A single knowledge base of relationships, consisting of Davenport sign links, can serve as the underlying organizing principle and data resource for any number of documents, and for the Davenport entry links found in all the tables of contents, indexes, and glossaries of those documents. Such a knowledge base is conceptually independent of particular documents; it does not point at the content of any document, and it is wedded to a field of knowledge, rather than to particular documents that belong to that field. Such a knowledge base could conceivably categorize and cross-reference a body of knowledge so thoroughly that it could serve as a primary resource for a wide variety of documents that contain information relevant to that body of knowledge.

[27] One way to make the index more self-consistent is to have all the index commands inserted and maintained by an indexing specialist, who can begin by designing what amounts to a list of topics and subtopics and a web of relations between them (a "navigational knowledge base"), and who then inserts the appropriate indexing commands, together with the full index-level addressing information, into the text wherever material related to the topics in the nodes appears. The services of indexing specialists are often used, but their use in technical documents has some serious drawbacks:

1. It is not reasonable to expect an indexing specialist to understand either the content of the document, or the needs of the users of the document, much less both, well enough to create a navigational knowledge base that will be more than minimally useful.
2. Typically the design of the navigational knowledge base is communicated poorly (if at all) to those who maintain the document, and those who maintain the document have little influence over the design or further development of that navigational knowledge base. Often, the navigational knowledge base is not kept with the document; its existence is merely implicit in the actual entries in the index, in the table of contents, and possibly in the glossary.
3. Considerable effort is wasted because, in order to meet the needs of the batch process that will create the index, aspects of the navigational knowledge base (perhaps represented as "index levels") must be spelled out completely in the indexing markup that must be inserted wherever the topic occurs in the document, and it often occurs in many places. The opportunity for introducing navigational knowledge base inconsistencies and other errors is huge. The inherent redundancy also makes it more expensive to provide all the cross-indexing that might be useful. It also tends to make the source form of a document difficult to maintain, because it becomes so laden with indexing markup that it becomes difficult to read for understanding.
4. The problem of eliminating redundancies and inconsistencies from concurrent aggregated master indexes of collections of documents that were originally indexed without knowledge of each other is very difficult.

By contrast, for any particular document or set of documents, there must be "webs" (i.e., sets of hyperlinks) that perform quite very specific navigational functions (e.g., tables of contents, indexes, and glossaries) that are wedded to the documents, rather than to the field of knowledge to which the documents belong.

With a web of sign links, a consistent and well-ordered view of a field of knowledge can be rationalized, expressed, and maintained.[28] A web of entry links can be used to connect the nodes in the web of sign links with the appropriate portions of particular documents.

4.4 Sign Links

Elements conforming to the Davenport **sign link** (`dav.sign.ilink`) architectural form are used recursively to construct webs of topics and subtopics. Each instance establishes a knowledge-bearing relationship between some anchor[29] and some other anchor.[30] In order to allow hierarchies of `sign`s, an anchor may recursively consist of one or more `sign`s.[31]

The recursive sign link mechanism is sufficiently flexible that, for example, multiple sign links throughout some set of navigational knowledge bases can use a single other sign link node in any one of them (or even in some other navigational knowledge base) as a subtopic of all of them.[32]

The purpose of a sign link is not primarily to represent a potential for traversal; Davenport **entry links** (see Section 4.5) are to be used for that purpose. The purpose of a sign link is to represent an epistemological relationship; if it is used directly for navigation, such use is not defined by this standard.[33]

This form has a content model of `%HyBrid;`, a HyTime-defined wild-card parameter entity;[34] the model allows a wide variety of content and hence maximizes the possible uses to which `sign` links can be put, and the flexibility available to Davenport document architects.

The `davrole` attribute's value can be used to indicate that the sign link is intended to be referenced by entry links in tables of contents (`toc`), indexes (`index`), and/or glossaries (`glossary`).[35]

---

[28] "Consistent" here may really mean "consistently inconsistent" in cases where there are competing schools of thought whose disparate views must all be accomodated and represented.

[29] In HyTime jargon, an *anchor* is any information object, such an an SGML element, that actually serves as one linkend of a link. In the common vernacular of hypertext, however, an *anchor* is any information object that could serve as a linkend; in HyTime jargon the latter is called a *potential anchor*.

[30] For example, in an aircraft manual, a sign link may represent a term-definition relationship between the term "pitot-static tube" and the definition "part of a sensing device used to determine the airspeed of aircraft in flight."

[31] This allows many levels of recursion and the representation of arbitrarily complex webs of relationships. For example, an information object consisting of the word "pitot" could be linked via an item-to-related-item sign link to an aggregate set of related sign links. One of these related sign links might signify a portion-whole relationship between "pitot-static tube" and "airspeed indicator." One might signify an item-description relationship between "pitot-static tube" and "little hook-like piece of metal tubing hanging from the underside of the left wing." Yet another might be an item-to-related-item link between "pitot-static tube" and "Henri Pitot, d. 1771," and this link might also serve as one end of yet another item-to-related-item link between it and the entire field of fluid dynamics.

[32] In other words, webs of sign links need not be strictly hierarchical; they could be metaphorically described as "arbitrary directed graphs," or, perhaps more colorfully, as "bowls of spaghetti."

[33] Obviously, one can imagine powerful applications, however, that could be used to browse webs of sign links, thus providing "secret passageways" or "back channels" between entry links.

[34] See the HyTime standard for details about `%HyBrid;`.

[35] The terms in parentheses are Davenport-defined SKNs for `davrole`, a list of which may be found in Section 4.2.1.

The `anchrole` attribute indicates the semantic roles played by the linkends.[36] Section 4.4.1 defines Davenport semantic key names (SKNs) for use in this attribute. If it is desired that either or both linkends have the potential to be an aggregate anchor, a "`#AGG`" should follow the SKN corresponding to the anchor in question.

The **anchrole semantic role definition generic identifiers** (`asrdgis`) attribute specifies the generic identifiers of the `dav.srdef`-form elements that define the SKNs used in the value of the `anchrole` attribute. If the `asrdgis` attribute is not specified, the Davenport built-in SKNs for use in `anchrole` attributes may be used, plus any SKNs defined by `DavASRD`-type elements in the document instance.

(The semantics and syntax of the remaining attributes are described in the HyTime standard.)

```
                    <!-- Sign Link -->
<!element dav.sign.ilink -- Basic building block for knowledge base
                            representation; a node in a knowledge base.
                            Expresses a relationship between a pair of
                            (possibly aggregate) anchors. --
                         - O  (%HyBrid;)* >
<!attlist dav.sign.ilink
    Davnport    CDATA    #FIXED       dav.sign.ilink
    davrole     -- possible roles: toc, index, glossary, lof, lot --
                CDATA    #IMPLIED  ""  -- Default: no defined semantic role --
    HyTime      NAME     #FIXED    ilink
    id          ID       #REQUIRED     -- see HyTime standard --
    anchrole    -- can be #FIXED; could be one of the following pairs of SKNs
                   (or define more using dav.srdef-form elements).  Either
                   or both SKNs could be characterized as potential aggregates
                   in the usual HyTime fashion (by using #AGG immediately
                   following the affected SKN). --
                        symbol          value
                        item            descript
                        item            idItem
                        item            eqItem
                        item            simItem
                        item            oppItem
                        item            relItem
                        instance        class
                        subset          superset
                        portion         whole
                        owner           possessn
                        synopsis        expansn --
```

[36]This is a required HyTime attribute, and is meant lexically to be a name list; the HyTime lexical specification is `lextype((NAME, s+, (RNI, "AGG")?), (s+, NAME, s+, (RNI, "AGG")?)+)`; see the HyTime standard for details, and for a discussion of the use of "`#AGG`" to specify aggregate anchors.

```
               -- HyTime lexical constraint:
                  lextype((NAME, s+, (RNI, "AGG")?),
                  (s+, NAME, s+, (RNI, "AGG")?)+) --
               CDATA    #REQUIRED
    asrdgis    -- One or more generic identifiers of elements used to
                  define the SKNs that appear in the value of the anchrole
                  attribute. --
               CDATA    #IMPLIED      "DavASRD" -- Default: all SKNs
                  defined for anchor roles by this standard and all
                  SKNs defined in the document instance using DavASRD-type
                  elements may appear in the anchrole attribute. --
    linkends   IDREFS    #REQUIRED           -- see HyTime standard --
    extra    -- lextype(("E"|"I"|"A"|"N"|"P"), (s+, ("E"|"I"|"A"|"N"|"P"))*) --
               NAMES    #IMPLIED             -- see HyTime standard --
    intra    -- lextype(("E"|"I"|"A"|"N"|"P"), (s+, ("E"|"I"|"A"|"N"|"P"))*) --
               NAMES    #IMPLIED             -- see HyTime standard --
    endterms   IDREFS    #IMPLIED            -- see HyTime standard --
    aggtrav  -- lextype (("AGG"|"MEM"|"COR"), (s+, ("AGG"|"MEM"|"COR"))*) --
               NAMES    #IMPLIED    agg      -- see HyTime standard -->
```

4.4.1 Built-in Anchor Roles (for anchrole of Sign Link)

The element type DavASRD is defined and instantiated in this Standard in order to provide built-in Davenport anchor semantics for SOFABED applications. (See Section 4.2 for a discussion of the dav.srdef architectural form on which the DavASRD element type is based.)

```
<!ELEMENT DavASRD  -- Davenport-defined built-in semantic role
                      definition for use with anchrole --
                   - O   (#PCDATA) >
<!ATTLIST DavASRD
    Davnport   CDATA    #FIXED        dav.srdef
    skn        -- names to be used as the value of a HyTime anchrole
                  attribute, representing the semantic described
                  in the content of this element, and/or the combination
                  of SKNs specified by the aliasfor attribute. --
               CDATA    #REQUIRED
    roletype   (davrole|anchrole|any) #FIXED anchrole
    aliasfor   CDATA    #IMPLIED      ""  -- default: none --
    alia4gis   CDATA    #IMPLIED      "DavDSRD DavASRD" -->
```

The following are definitions of pairs of names that can appear in the value of the anchrole attribute of element types of dav.sign.ilink (see Section 4.4) form and of non-Davenport HyTime ilink form, as well as in the aliasfor attribute of other dav.srdef-form elements with matching `roletype` attribute values. These SKNs are built into all SOFABED-compliant applications. For the sake of clarity, these definitions strongly imply that the order of the linkends as specified here is a requirement; however, that is not the case. The order of linkends specified in the `linkends` attribute must correspond to the order of the names given in the `anchrole` attribute, but they can appear in a different order than the order specified below.[37]

```
<DavASRD skn="symbol value">
A "signifying" relationship.  The purpose of the first anchor is to
signify the other ("value") anchor.  (The first anchor might be, e.g.,
a term, a title, or a heading.) The purpose of the second anchor is to
show what the other ("symbol") anchor signifies.  This second anchor
might be, e.g., a definition for the term serving as the other anchor,
or a chapter whose title serves as the other anchor, or a paragraph
whose heading serves as the other anchor.  Usage: element types of
dav.sign.ilink form.

<DavASRD skn="item descript">
A "descriptive" relationship.  The first anchor represents something
to be described, and the second anchor is its description. (Try not to
confuse the semantic of the item-description anchrole pair with the
symbol-value anchrole pair!)  Usage: element types of dav.sign.ilink
form.

<DavASRD skn="item idItem">
An "identity" relationship.  The first anchor is, in some sense(s),
exactly the same as the second anchor, whatever either may be. Usage:
element types of dav.sign.ilink form.

<DavASRD skn="item eqItem">
An "equivalence" relationship.  The first anchor can, in some sense or
under some circumstance, be substituted for the other anchor, whatever
either may be.  Usage: element types of dav.sign.ilink form.

<DavASRD skn="item simItem">
A "similarity" relationship.  The first anchor is somehow like the
other anchor, whatever either may be.  Usage: element types of
dav.sign.ilink form.

<DavASRD skn="item oppItem">
An "opposition" relationship.  The first anchor is in some sense the
exact opposite of the second anchor.  Usage: element types of
```

[37] For example, in the case of a `symbol-value` sign link, the order can be either (symbol, value) or (value, symbol).

dav.sign.ilink form.

<DavASRD skn="item relItem">
A "generic" relationship. The first anchor is somehow related to the other anchor, whatever either may be. This is a vague, undefined relationship, which should be used only if no other Davenport anchrole SKN name-pair is appropriate. Usage: element types of dav.sign.ilink form.

<DavASRD skn="instance class">
An "instance-class" relationship. The first anchor constitutes one or more examples of an entire class; the second anchor is a representation of that class. Usage: element types of dav.sign.ilink form.

<DavASRD skn="subset superset">
A "subset-superset" relationship. The first anchor is in some sense a subset of the set of things which serves as the other anchor. The second anchor (superset) may or may not constitute the set of all things of some class; there is no implication that "all" of some set must serve as the superset anchor. Usage: element types of dav.sign.ilink form.

<DavASRD skn="portion whole">
A "part-whole" relationship. The first anchor is in some sense a part of the second anchor, which has some sort of completeness which the other anchor lacks. Usage: element types of dav.sign.ilink form.

<DavASRD skn="owner possessn">
A "possessive" relationship. The first anchor (e.g., an individual or a corporation) has some proprietary right, title, and/or interest in the second anchor. Usage: element types of dav.sign.ilink form.

<DavASRD skn="synopsis expansn">
A "synopsis-expansion" relationship. The first anchor is a summary or abstract of the second anchor, which is longer and more elaborately detailed. It could be, e.g., the complete, uncut, or original version whose digested form appears as the other anchor, or it could be a version of the other anchor which is longer because annotations and/or commentary have been added. Usage: element types of dav.sign.ilink form.

4.5 Entry links

Each entry in a navigational table (i.e., a table of contents, an index, or a glossary) may have pointers to places in the document(s) where appropriate material appears. In on-line documents, these pointers may be hidden from the user, because they are directly used only by the browsing application whose responsibility it is resolve the pointers and effect the traversal, if the user demands it. In paper documents, these pointers usually take the form of page numbers, and the user is responsible for effecting the traversal by physically finding the indicated pages by the numbers printed on them. In each kind of navigational table each entry is a hyperlink that expresses a relation between some categorizing or characterizing symbol(s), word(s), phrase(s), or title(s), on the one hand, and some other portion(s) of the document, on the other. In Davenport, such entries are represented as **entry links** (`dav.entry.ilink`).

An `entry` link always has exactly two linkends, whose anchor roles are `topic` and `target` respectively.

The `topic` is something that indicates the reason why a user might want to traverse the link to the `target` linkend. There is no requirement that the `topic` anchor be some set of (one or more) `sign` links.[38] If the `topic` anchor is an aggregate (i.e., a HyTime `nameloc`-type element which points to more than one document location), and if the `davrole` attribute of that `nameloc`-type element specifies `topcntxt` (see Section 4.2.1), the lexically last listed item in the aggregate is the specific topic of the `entry`, and the other document locations at which the `nameloc`-type element points are used to establish an ordered topical context for that last-listed topic.[39] An aggregate `topic` anchor that includes `sign` links should list the `sign` links in the order in which those `sign` links link themselves to one another. (To do otherwise would be to fail to reflect a presumably valid relationship between the `sign` links in the knowledge base.)[40]

The `target` linkend is some set of information objects in one or more documents, to which users

[38]However, if there is any possibility that an index will someday have to be merged with other indexes, the disciplined creation and use of `sign` links in a knowledge base of `sign` links is an option that should be considered. See Section 4.3.1.

[39]Levels of indexing can be specified in this way. For example, an aggregate `nameloc` with `davrole=topcntxt` can be used with `sign` links to establish the context in which a topic is listed in an index. Such use will have the effect of making the topical context unambiguous even when the topic is specified by a `sign` link that serves as a link end (a subtopic) of more than one other `sign` link.

[40]Some examples:

- In a table of contents, the `topic` anchor of an `entry` should be the title of a section, while the `target` anchor should be the section itself.
- In an index `entry`, the `topic` anchor of an `entry` might be an aggregate of an item-to-related-item `sign` link between "pitot-static tube" and "Henri Pitot, d. 1771," preceded by an item-to-related item `sign` link between it and "fluid dynamics." This is how to establish that our example `entry` requires that "pitot-static tube" and "Henri Pitot, d. 1771" be listed under the topic of "fluid dynamics," rather than under the the topic of "pitot," which also uses the same `sign` link as one of its anchors. (See footnotes 30 and 31 for details about the context of the foregoing example.)
- In a glossary, an `entry`'s `topic` anchor shold be a symbol-value type of `sign` link, and its `target` should be one or more places in the document(s) where the term is used.

who wish more information on a `topic` may traverse.

This form has a content model of `%HyBrid;`, a HyTime-defined wild-card parameter entity;[41] the model allows a wide variety of content and hence maximizes the possible uses to which `entry` links can be put, and the flexibility available to Davenport document architects.

```
                     <!-- Entry Link -->
<!element dav.entry.ilink  -- Basic navigational link.  It links node(s) in a
                              knowledge representation (e.g., a network of
                              sign links) to anchors in the document(s). --
           - O  (%HyBrid;)* >
<!attlist dav.entry.ilink
    Davnport    CDATA    #FIXED        dav.entry.ilink
    davrole     -- possible roles: toc, index, glossary, lof, lot --
                CDATA    #IMPLIED  ""  -- Default: no defined semantic role --
    HyTime      NAME      #FIXED    ilink          -- see HyTime standard --
    id          ID        #REQUIRED                -- see HyTime standard --
    anchrole    CDATA     #FIXED "topic #AGG target #AGG"
                                                   -- see HyTime standard --
    linkends    IDREFS    #REQUIRED                -- see HyTime standard --
    extra    -- lextype(("E"|"I"|"A"|"N"|"P"), (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                NAMES    #IMPLIED                  -- see HyTime standard --
    intra    -- lextype(("E"|"I"|"A"|"N"|"P"), (s+, ("E"|"I"|"A"|"N"|"P"))*) --
                NAMES    #IMPLIED                  -- see HyTime standard --
    endterms    IDREFS    #IMPLIED                 -- see HyTime standard --
    aggtrav    -- lextype (("AGG"|"MEM"|"COR"), (s+, ("AGG"|"MEM"|"COR"))*) --
                NAMES              agg             -- see HyTime standard -->
```

4.6 Notes and Special Provisions for Pre-HyTime Systems

`entry` links and `sign` links may contain their own linkends. As in any HyTime-compliant `ilink`-form element type, the `linkends` attribute of an `entry` link must have a value consisting of the unique identifiers of the elements that serve as the ends of the link. However, a more constrained form of DASH-compliant document is also possible, in which the elements that serve as linkends, *in addition* to being pointed to by the (required) `linkends` attribute, can also be contained within the link element. This optional method of representing hyperlinks is redundant, in that the `linkends` attributes of the link elements point to the content of those same link elements, but it has the advantage of allowing the construction and use of HyTime-compliant documents by means of certain existing tools, in advance of the general availability of HyTime-compliant applications for document creation, manipulation, and presentation. Such documents will be useful immediately, and they will continue to be useful after compliance with HyTime is widespread and commonplace. Once HyTime processing is implemented, linkend elements will no longer need to be placed within

[41] See the HyTime standard for details about `%HyBrid;`.

their respective link elements; for example, new entries could be added to a glossary, for example, without any longer requiring that each new entry contain its own linkends. [42]

5 HyTime Support

There are two distinct sets of user requirements regarding indexes and glossaries that must be met by the SOFABED:

1. The ultimate requirement is that the SOFABED set forth conventions for representing indexes and glossaries (perhaps among other things) in "navigational" documents that are perhaps external to and distinct from the documents whose contents are to be navigated thereby. It must be possible to create such navigational documents without, as it were, the cooperation of the documents to be navigated; for that reason, among others, the representation of such navigational documents can be expected to require substantial HyTime support.

2. In order to meet current needs in advance of the availability of complete HyTime engines, a set of conventions is needed for representing index and glossary information within the same document as the information being indexed – for navigating only *within* a single SGML document.

In both cases, the essential structure can be identical, with the only difference being the degree to which various HyTime addressing facilities are involved.

For the case in which documents contain all their own navigational information, the minimum required HyTime support is declared as follows:[43]

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

In the case of external navigational documents, much more HyTime support may be in order, but the following declarations, at least, would be required:[44]

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
<?HyTime MODULE base exidrefs refctl>
<?HyTime MODULE locs>
<?HyTime MODULE links>
```

[42] In fact, in most situations, it is probably unnecessary even today to resort to the strategem of putting the linkends inside the links. HyTime independent links (`ilinks`), purely by themselves, do not require any special re-parsing or other HyTime-specific processing that is beyond the ability of SGML application substrates. The connection between an `ilink` and its linkends is expressed by giving each linkend element an SGML unique identifier, and by using those unique identifiers as the value of the `ilink`'s `linkends` attribute. It doesn't matter where the linkends are, or where the link is, so long as each linkend is an element that has a unique identifier, and so long as the link and all its linkends are contained in a single SGML document.

[43] The minimum required HyTime support for the SOFABED would not provide a variety of desirable so-called "SGML++" features of HyTime, such as reference control, nor would it provide the ability to aggregate multiple anchors into a single linkend.

[44] The `multloc` option of the `locs` module is also recommended.

In other words, the exidrefs option would have to be added to the base module, and the locs module would also be required. If the documents to be indexed are not completely cooperative, or if more of HyTime's validation features are desired, the following support declarations are recommended:

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
<?HyTime MODULE base exidrefs refctl>
<?HyTime MODULE locs multloc anydtd anysgml mixspace>
<?HyTime MODULE links manyanch>
```

In other words, to allow documents conforming to a variety of DTDs or SGML declarations to be indexed in a single master index document, that document's HyTime support declarations should list the anydtd and anysgml options of the locs module. For the sake of convenience, the mixspace option would permit any given HyTime nameloc-form element to contain either entity names or element names, or both.

6 Davenport Support

Since it relies on HyTime for its semantics, the SOFABED can afford to be simple; it need not have the complex array of modular support declarations that HyTime requires. For the present, anyway, it is only necessary that Davenport documents be recognized as conforming to the SOFABED. However, future versions of the DASH that support specialized forms of HyTime activity tracking and/or tabular data representation via HyTime FCSs may require more elaborate arrangements. Therefore, the SOFABED uses the same strategy for Davenport support declarations that is used in HyTime. This strategy gives future modules of the DASH the ability to add specialized architectural forms, while maintaining total backward compatibility and providing for future modularity.[45]

A document indicates conformance to the SOFABED by the APPINFO parameter of the SGML declaration, which specifies the name of the "Davenport Prefix" that identifies SGML processing instructions that represent "Davenport Declarations." Davenport declarations indicate the facilities required to support the document. The format is:

```
Davnport=name
```

if a prefix name is to be specified, otherwise "Davnport" alone. The prefix name must be a valid name in the concrete syntax of the SGML declaration.

A Davenport declaration is represented as an SGML processing instruction that begins with the Davenport prefix, followed by a "declaration name" and other parameters as required by the declaration type. Each of the parameters must be preceded by one or more "white space" characters: i.e., in SGML terms, RS, RE, and SPACE. White space characters can also follow the last parameter. The case of the parameters is not significant.

[45] These needs for future additions, backward compatibility, and modularity explain why there is, in the SOFABED at least, "so much dressing for so little salad."

A Davenport document includes in its prolog, along with appropriate HyTime support declarations, one or more "Davenport support declarations" that collectively identify all of the modules and options for which support is required in the document. The same requirements that the HyTime standard makes with regard to HyTime support declarations and how they are handled by HyTime applications also apply to Davenport support declarations. A Davenport support declaration is represented as an SGML processing instruction whose declaration name is "`VERSION`" or "`MODULE`." If the name is "`MODULE`," it must be followed by one of the module names in the following list,[46] thereby declaring that support is required for that module:

`sofabed`

The module name must be followed by the names of the module's optional facilities (if any) for which support is being declared. For example, using the SGML reference concrete syntax,

```
<?Davnport  VERSION "DASH:1993">
<?Davnport  MODULE sofabed  option1  option2>
```

(However, there are currently no optional facilities in the SOFABED.)

The Davenport SOFABED Module (`sofabed`) requires support of the HyTime standard at some one of the levels described in this Section, as well as support of the facilities described in Section 4 of this Standard.

[46] Because the SOFABED is the first module of the DASH to be completed, there is only one item in the list at the time of this writing.