Davenport Group Draft Advisory Standard (DGDAS)

Editors: Steven R. and Victoria Taussig Newcomb*

Version 0.2.1, 10 May 1992

Contents

1 Introduction **2**

2 References and Resources **4**

3 How to Read the DGDAS **4**
3.1 Editorial Conventions 5
3.2 Meta-DTD Conventions 5
3.2.1 Architectural Form Naming Conventions 5
3.2.2 Attribute List Conventions 6
3.2.3 Content Model Conventions 6

4 Davenport Architectural Forms **7**
4.1 General-purpose Architectural Forms 7
4.1.1 Davenport Document (`dav.document.HyDoc`) 7
4.1.2 On-site Location (`dav.siteLocation.HyBrid`) 8
4.1.3 Choice Availability Signifier (`dav.choiceSignifier.HyBrid`) 9
4.1.4 Dyad (`dav.dyad.HyBrid`) and Dyad link (`dav.dyad-link.ilink`) 9
4.1.5 Subject (`dav.subject.HyBrid`) 10
4.1.6 Predicate (`dav.predicate.HyBrid`) and Predicate link (`dav.predicate-link.ilink`) 11
4.2 Special-purpose Architectural Forms 12

*This version incorporates editing instructions given at the Davenport Workshop held 22-25 March, 1992, at Hewlett-Packard, Fort Collins, Colorado. Representatives of Avalanche, Electronic Book Technologies, Hewlett-Packard, IBM, Novell, O'Reilly & Associates, Open Software Foundation, Open Systems Solutions, TechnoTeacher, and Unix System Laboratories participated in the Workshop. 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 version incorporates revisions necessitated by the editorial changes made to the International Standard version of HyTime (ISO/IEC IS 10744:1992, May-June, 1992). Publication of the final, adopted International Standard version of HyTime by the ISO and its national member bodies is anticipated in June, 1992.

4.2.1 Index collection (`dav.index.elemloc`) 12
4.2.2 Glossary Set (`dav.glossary-set.ilink`) 14

5 Support Declarations **15**

6 Informative Annex: Sample Davenport Document **16**
6.1 DTD for Sample Davenport Document 16
6.2 Sample Davenport Document Instance 21

List of Figures

1 *Diagram of the index and glossary in a hypothetical Davenport-compliant document.* 23

1 Introduction

The Davenport Group Draft Advisory Standard (DGDAS) defines a model and language for the representation of online and printed technical documentation of the kind usually supplied by Unix and other open systems vendors to their customers. The DGDAS will allow such 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 independently. The DGDAS will allow documents of dissimilar SGML document types[1] 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, 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 DGDAS makes it possible for the same original document to be formatted for online delivery as hypermedia as well as for printing on paper.

The DGDAS is being developed partly in response to the widespread sentiment among software and systems vendors, and those who supply and maintain their documentation, that a standard representation of hypertext links – hyperlinks – should be adopted by all, and that standardizing such a representation 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
- the increased usefulness of information that can access a broad base of other documents by hyperlink improves its own marketability and its publisher's profits, even when other publishers may receive additional revenue when such links are traversed by users.

[1] 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.

The DGDAS 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 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 DGDAS architectural forms are based on, and are entirely consistent with, HyTime.[2]

Aside from hyperlinks and facilities for addressing information objects (facilities that support hyperlinks), other candidates for standardization in Davenport-compliant documents include:

- representation of publishers' policies with regard to links to their documents and wholesale inclusions of portions of their documents in documents published by others. This would facilitate royalty accounting, and it would provide the infrastructure 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.

- 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 information that is essential to the meaning of the information that constitutes a multimedia presentation, for example, and 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.

When performing the semantic processing required by the DGDAS 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 that the Davenport Engine will apply additional constraints in accordance with the prescriptions of particular Davenport architectural forms. Therefore, a Davenport Engine need not duplicate the functionality of a HyTime Engine, but could instead consist of a set of validating routines added as a module to a system that already contains the appropriate subset of the functionalities of a HyTime Engine.[3]

Like the HyTime Draft International Standard on which it is based, this Davenport Group Draft Advisory Standard is intended to be used as an infrastructure for platform-independent information

[2]HyTime is the International Standard Hypermedia/Time-based Structuring Language defined by ISO/IEC IS 10744:1992.

[3]Some of the features of HyTime, such as the ability to schedule events in virtual or real time and/or space, are not presently used in this draft. 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 multimedia 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.

interchange; it is *not* intended for encoding the internal representation of information on which specific hypertext application programs act. 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 an application is defined neither in HyTime nor by Davenport.

2 References and Resources

1. ISO 8879, *Standard Generalized Markup Language* (SGML). This is the international standard metalanguage on which HyTime and DGDAS are built.

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

3. ISO 9070, *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. The DGDAS may ultimately become "public text" by the method outlined in ISO 9070.

4. ISO IS 10744, *Hypermedia Time-based Structuring Language* (HyTime). Itself founded on SGML, HyTime is the standard foundation on which the DGDAS is built. (See footnote 2.)

5. The *SGML SIGhyper Newsletter* is published by the SGML Users' Group Special Interest Group on Hypertext and Multimedia. It publishes examples of HyTime documents and other information relevant to the DGDAS, including copies of the HyTime IS itself. Contact SGML SIGhyper, c/o TechnoTeacher, Inc., 1810 High Road, Tallahassee, Florida 32303 USA; tel: +1 904 422 3574; fax: +1 904 386 2562; Internet: SIGhyper@techno.com.

3 How to Read the DGDAS

The DGDAS is expressed using a formalism that was first used to represent the HyTime IS: the so-called "meta-DTD." Rather than containing the `ELEMENT` and `ATTLIST` declarations normally found in SGML document type definitions (DTDs), the DGDAS sets forth templates, or "SGML architectural forms," which are intended solely for human readers, and which are not meaningful to SGML parsers. SGML architectural forms are, in essence, rules for constructing element types, including attribute definitions and content model constraints. The grammar of SGML architectural forms resembles the grammar usually used to express SGML document type definitions; the 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 components.)

3.1 Editorial Conventions

Section 4 consists of normative definitions and descriptions of the Davenport Architectural Forms (the SGML templates) of the DGDAS. Each architectural form is normatively defined using the formalism developed to express the HyTime meta-DTD; this is followed by a description of the form in English.[4]

An illustrative example of a hypothetical SGML/HyTime/Davenport DTD and document instance appears in the Annex (16).[5] It is meant to help readers understand how Davenport forms may be integrated with HyTime forms in the building of an actual DTD from the Davenport and HyTime meta-DTDs.

3.2 Meta-DTD Conventions

As in the HyTime IS, architectural forms are specified using element type forms and attribute list type forms.[6] Certain conventions have been followed in these declarations to clarify the element content models and the attribute definitions and their data types. 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 Architectural Form Naming Conventions

The name of each Davenport Architectural Form consists of two or three fields, with a period (.) used as the field separator.

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. It consists of one or two subfields; if there are two subfields, they are separated by a dash (–). If there is only one subfield, the architectural form has general applicability (e.g. `choiceSignifier`). If there are two subfields, the first field is a category of structure within a Davenport document, e.g. `glossary`. The second subfield indicates the function of the architectural form within the category, e.g. `set` (together, `glossary-set`).

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

[4] For a thorough understanding of each Davenport architectural form that is based on a HyTime architectural form, it is imperative to consult the HyTime IS. The DGDAS is not meant to recapitulate the HyTime IS.

[5] In standards jargon, this part of a standards document is characterized as *informative* and is understood as *not* forming a part of the standard itself. The parts of the standard with which conforming applications are required to comply are called *normative*.

[6] The DGDAS does not define any Davenport attribute list type architectural forms, but future versions may do so.

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. Since the attributes defined for these forms will, unless explicitly aliased, keep the names given them in the DGDAS, their names will be no longer than a universally parseable eight characters.

3.2.2 Attribute List Conventions

Components contributed by HyTime architectural forms to Davenport architectural forms are identified by associated comments. Constraints on attribute values and data types imposed by HyTime or by the DGDAS are also indicated by comments.

Common attributes 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 *ad libitum* to Davenport-compliant element declarations.

3.2.3 Content Model Conventions

Content models for Davenport architectural forms are shown as ordinary SGML content model declarations. The content models may be further defined by associated 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. 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.

- 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.

- The content model `EMPTY` in an architectural form would mean that a conforming element type may not contain anything; it is already fully constrained.

4 Davenport Architectural Forms

In addition to the architectural forms defined in this document, the DGDAS uses the HyTime document location addressing and linking constructs; Davenport applications will need to provide support for much of the HyTime Location Address Module, the HyTime Hyperlinks Module, as well as the Base and Measurement Modules (see Section 5 and the HyTime IS for details).

4.1 General-purpose Architectural Forms

```
<!-- Entities declared for use in defining
     content models of Davenport architectural forms. -->
<!ENTITY % loc     -- HyTime location address architectural forms --
     "bibloc | elemloc | entloc | strloc | tokenloc | treeloc |
     pathloc | attloc | notloc">
<!ENTITY % link    -- HyTime hyperlink architectural forms --
     "clink | ilink">
<!ENTITY % resbase -- HyTime Base Module forms usable as resources --
     "HyLex">
<!ENTITY % HyBrid  -- Architectural forms allowed in HyBrid element types --
     "HyBrid | nHyTime | #PCDATA">
<!ENTITY % davdyad -- Davenport dyad forms --
     "dav.dyad | dav.subject.HyBrid | dav.predicate.HyBrid">
<!ENTITY % davlink -- Davenport link forms --
     "dav.glossary-link.ilink | dav.glossary-set.ilink">
<!ENTITY % davincl -- Davenport inclusions --
     "dav.siteLocation.HyBrid | dav.choiceSignifier.HyBrid
     | dav.index.elemloc | %davlink; | %loc; | %link;">
```

These parameter entities are declared in order to make the specification of content meta-models easier. Some include HyTime forms, the definitions and descriptions of which should be sought in the HyTime IS. There are no Davenport architectural forms in this version of the DGDAS comparable to the HyTime `HyBrid` and `nHyTime` forms; non-Davenport elements (which may or may not be HyTime elements) may be used wherever `%HyBrid;` is specified in the content model of any HyBrid-type Davenport elements. If these HyBrid elements themselves contain Davenport elements, those sub-elements will always (in this version of the DGDAS) receive the appropriate Davenport processing.

4.1.1 Davenport Document (`dav.document.HyDoc`)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.document.HyDoc - O  -- Davenport document element --
    (%HyBrid; | %resbase; | %davdyad;)*  +(%davincl;)>
<!attlist dav.document.HyDoc
    Davnport  CDATA   #FIXED "dav.document.HyDoc"
```

```
    HyTime    NAME    #FIXED HyDoc
    id        ID      #IMPLIED >
```

The **Davenport document** (`dav.document.HyDoc`) architectural form conforms to the HyTime document element architectural form `HyDoc`. It defines an environment in which Davenport processing will take place for those sub-elements with a defined `Davnport` attribute, and where HyTime processing will take place for those elements (some of which will also be Davenport elements) with defined `HyTime` attributes (unless the value is `nHyTime`).

4.1.2 On-site Location (`dav.siteLocation.HyBrid`)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.siteLocation.HyBrid  - -  ANY    -- A potential anchor
        marked in context.  The content must be valid
        in the context of the dav.siteLocation.HyBrid. -->
<!attlist dav.siteLocation.HyBrid
    Davnport  CDATA   #FIXED   "dav.siteLocation.HyBrid"
    HyTime    NAME    #FIXED   HyBrid
    -- Allows content to include processable HyTime elements, and allows
       HyTime engines to recognize the endterm attribute. --
    id        ID      #REQUIRED
    context   (context | ncontext) #FIXED context
    -- Content would still be valid if markup were removed. --
    reftype   CDATA   #FIXED  "endterm dav.choiceSignifier.HyBrid"
    -- endterm attribute's value must be ID of choiceSignifier element --
    endterm   IDREF   #IMPLIED
    -- Allows a cue to the user to be associated with this element.  The
       cue can be used, in the context of hyperlink processing, to
       indicate that traversal to this anchor is possible.  In the
       HyTime IS, endterm is a "common attribute" that can be added to
       any element's attribute list. -- >
```

Davenport defines an **on-site location** element type form, `dav.siteLocation.HyBrid`, which provides a Davenport-standard way to point to fragments of content consistent with the common practice of marking places in a manuscript to which an index entry, for example, will refer. A `dav.siteLocation.HyBrid`-form element would normally be used to apply a unique identifier to something which would not otherwise be an element, and which therefore could not have a unique identifier. The content of a dav.siteLocation.HyBrid must be valid for the context in which the dav.siteLocation.HyBrid appears; an ordinary HyTime application will check whether this is so because the value of the HyTime common attribute `context` is fixed in this DGDAS as "context". The validation of this rule (that the content must be valid in the surrounding context) is based on a test consisting of removing the `dav.siteLocation.HyBrid` markup and testing the resulting

surrounding element for congruence with the DTD.[7]

4.1.3 Choice Availability Signifier (dav.choiceSignifier.HyBrid)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.choiceSignifier.HyBrid - - ANY
    -- signifier (hypertext button, index or toc entry, etc.).  In
       Davenport documents, the value of the HyTime common "endterm"
       attribute, if used, must be the ID of an element of this form. -- >
<!attlist dav.choiceSignifier.HyBrid
    Davnport    CDATA    #FIXED   "dav.choiceSignifier.HyBrid"
    HyTime      NAME     #FIXED   HyBrid
    id          ID       #REQUIRED
    conloc      IDREF    #CONREF >
```

HyTime does not define any particular structure (i.e., architectural form) for elements that contain information intended to indicate to the users of hyperdocuments that some choice is possible, for instance that it is possible to traverse to a particular document location that serves as an "anchor." The DGDAS declares a Davenport architectural form for this purpose, **choice availability signifier** (dav.choiceSignifier.HyBrid), whose content model is left to the discretion of the DTD designer. The purposes of dav.choiceSignifier.HyBrid are merely to identify element types that are intended to serve as endterms to Davenport applications for document validation purposes.

4.1.4 Dyad (dav.dyad.HyBrid) and Dyad link (dav.dyad-link.ilink)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.dyad.HyBrid  - O   -- subject/predicate (etc.) relation --
    (dav.subject.HyBrid, (dav.predicate.HyBrid | dav.predicate-link.ilink))>
<!attlist dav.dyad.HyBrid
    Davnport    CDATA   #FIXED   "dav.dyad.HyBrid"
    HyTime      NAME    #FIXED   "HyBrid"
    id          ID      #IMPLIED >

<!-- Davenport Architectural Form.  Normative. -->
<!element dav.dyad-link.ilink  - O  (#PCDATA) >
<!attlist dav.dyad-link.ilink
    Davnport CDATA    #FIXED  "dav.dyad-link.ilink"
    HyTime   NAME     #FIXED  ilink
    id       ID       #IMPLIED
```

[7] It is therefore important to avoid declaring element types that both (a) conform to the dav.siteLocation.HyBrid architectural form and (b) have structural significance which would make their content invalid in their surrounding context.

```
lexmodel NAMES    #FIXED  HyLex
lextype  CDATA    #FIXED  "#CONTENT idr2 linkends idr2"
                          -- if there is content, it must be a pair
                             of valid IDs; if the linkends attribute
                             is used, its value must be a pair of
                             valid IDs --
reftype  CDATA    #FIXED  "#CONTENT dav.dyad.HyBrid #SEQ
                           -- if there is content, the sequence of IDs
                              must satisfy the content model of
                              dav.dyad.HyBrid --
                           linkends dav.dyad.HyBrid #SEQ
                           -- if the linkends attribute is used, the
                              sequence of IDs must satisfy the content
                              model of dav.dyad.HyBrid --
                           endterm dav.choiceSignifier.HyBrid
                           endterms dav.choiceSignifier.HyBrid"
endterm  IDREF    #IMPLIED
linkends IDREFS   #CONREF
extra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                    (s+, ("E" | "I" | "A" | "N" | "P"))*) --
         NAMES    #IMPLIED
intra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                    (s+, ("E" | "I" | "A" | "N" | "P"))*) --
         NAMES    #IMPLIED
endterms IDREFS   #IMPLIED
aggtrav  -- lextype (("agg" | "mem" | "cor"),
                     (s+, ("agg" | "mem" | "cor"))*) --
         NAMES    agg
context  (context | ncontext) ncontext >
```

Many common information structures can be seen as being composed of dyadic components. Glossaries are sets of pairs of terms-to-be-glossed and their respective glosses; "subject-predicate" type pairs are also collected in quick-reference charts, dictionaries, thesauri, and even relational databases.[8] Together with the dyad, subject, and predicate forms, **dyad link** (`dav.dyad-link.ilink`) can be used to build complementary or interlocking element types to describe any of these kinds of information structures.

4.1.5 Subject (`dav.subject.HyBrid`)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.subject.HyBrid  - O  (%HyBrid;)* >
<!attlist dav.subject.HyBrid
```

[8] It is possible as well to analyze complex tables as sets of nested dyads.

```
    Davnport    CDATA   #FIXED   "dav.subject.HyBrid"
    HyTime      NAME    #FIXED   HyBrid
    id          ID      #IMPLIED
    reftype     CDATA   #FIXED   "endterm dav.choiceSignifier.HyBrid"
    endterm     IDREF   #IMPLIED
    conloc      IDREF   #CONREF >
```

The Davenport **subject** (`dav.subject.HyBrid`) architectural form is designed to form half of a subject-predicate dyad, the other half being the `dav.predicate.HyBrid` form. An element derived from this form may be used as the first of an ordered pair of anchors of a `dav.dyad-link.ilink`-type hyperlink.

4.1.6 Predicate (`dav.predicate.HyBrid`) and Predicate link (`dav.predicate-link.ilink`)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.predicate.HyBrid  - O
    (dav.dyad.HyBrid | dav.dyad-link.ilink | %HyBrid;)* >
<!attlist dav.predicate.HyBrid
    Davnport   CDATA   #FIXED    "dav.predicate.HyBrid"
    HyTime     NAME    #FIXED    HyBrid
    id         ID      #IMPLIED
    reftype    CDATA   #FIXED    "endterm dav.choiceSignifier.HyBrid"
    endterm    IDREF   #IMPLIED >

<!-- Davenport Architectural Form.  Normative. -->
<!element dav.predicate-link.ilink  - O  (#PCDATA) >
<!attlist dav.predicate-link.ilink
    Davnport CDATA    #FIXED  "dav.predicate-link.ilink"
    HyTime   NAME     #FIXED  ilink
    id       ID       #IMPLIED
    lextype  CDATA    #FIXED  "#CONTENT idrl linkends idrl"
                              -- if there is content, it must be a list
                                 of valid IDs; if the linkends attribute
                                 is used, its value must be a list of
                                 valid IDs --
    reftype  CDATA    #FIXED  "#CONTENT dav.predicate.HyBrid
                               -- if there is content, the IDs must
                                  conform to the content model of
                                  dav.predicate.HyBrid --
                              linkends dav.predicate.HyBrid #SEQ
                               -- if the linkends attribute is used, the
                                  IDs must conform to the content model
```

```
                              of dav.predicate.HyBrid --
                            endterm dav.choiceSignifier.HyBrid
                            endterms dav.choiceSignifier.HyBrid"
    endterm  IDREF    #IMPLIED
    linkends IDREFS   #CONREF
    extra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    intra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    endterms IDREFS   #IMPLIED
    aggtrav  -- lextype (("agg" | "mem" | "cor"),
                         (s+, ("agg" | "mem" | "cor"))*) --
             NAMES    agg
    context  (context | ncontext) ncontext >
```

The Davenport **predicate** (`dav.predicate.HyBrid`) and **predicate link** (`dav.predicate-link.ilink`) architectural forms are designed to allow element types to be built to accommodate single, multiple, or nested definitions, glosses, or other predicate-type information for any given subject item, for the making of glossaries, dictionaries, quick-reference tables, etc. An element derived from either form may be used as the second of an ordered pair of anchors of a `dav.dyad-link.ilink`-type hyperlink.

4.2 Special-purpose Architectural Forms

4.2.1 Index collection (`dav.index.elemloc`)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.index.elemloc - O (nmlist | nmidlst | nmquery)*
    -- Collects index locations; any location pointed to directly or
       indirectly by a dav.index.elemloc is considered to be indexable.
       Associates aggloc attributes (if there is more than one name in
       the ultimate list), and an optional endterm with the document
       location(s) to which it points. --
    -- The nmlist, nmidlst, and nmquery architectural forms are used as
       defined by HyTime. -- >
<!attlist dav.index.elemloc
    Davnport   CDATA   #FIXED     "dav.index.elemloc"
    colscope   -- defines the limits of the index by naming a parent or
                  parents for all the elements from which index items
                  are extracted for the name lists in content --
               IDREFS  #IMPLIED
    HyTime     NAME    #FIXED      elemloc
```

```
id          ID      #REQUIRED
-- HyTime aggloc attributes are below: these apply only if the name
   list made from the content has more than one entry.--
ordering    (ordered | noorder) noorder -- Default: Order of names is
                                        not significant. --
set         (set | notset)      set   -- Default: Please remove any
                                        duplicate names. --
agglink     (agglink | nagglink)  nagglink  -- Default: Please do not
                                               create a link between the
                                               aggregate and each member. --
endterm     IDREF   #IMPLIED
conloc      IDREF   #CONREF
-- If used, the conloc attribute refers to another element whose
   content "becomes" the content of this element. --
refrange    CDATA   #FIXED     "#ALL X endterm I"
-- The endterm information must be local to the document in which the
   locationHandle itself appears.  Everything else can be anywhere. --
reflevel    CDATA   #FIXED     "#ALL -1"
-- Do not abort resolution of document locations; go all the way to
   the end of each "location ladder." --
reftype     CDATA   #FIXED     "#ALL #ANY endterm dav.locationSignifier"
-- The endterm attribute, if used, must point to an element
   conforming to any dav.choiceSignifier.HyBrid architectural form.  With
   the sole exception of the endterm attribute, all attributes in the
   attribute list may refer to elements of any architectural form and
   generic identifier.  (This fixed attribute value almost certainly
   will be changed when used in any specific DTD, if for no other
   reason than to replace "dav.choiceSignifier.HyBrid" with some actual
   generic identifier.) -- >
```

The Davenport **index collection** (`dav.index.elemloc`) architectural form is built on the HyTime **element location address** (`elemloc`) architectural form, and it designates a collection of items to be indexed. In addition, `dav.index.elemloc` element types may point to information that will be shown to a user indicating that should the location be the anchor of a link, some kind of choice regarding traversal is available; this information is called the "linkend term" or "`endterm`" information in HyTime jargon.[9] Finally, several other pieces of information are also given with each `dav.index.elemloc` type element regarding properties of the aggregate if there are multiple locations involved, the permitted range of reference and of resolution, and the scope of the index collection.

The DGDAS `dav.index.elemloc` architectural form is in all respects identical to the HyTime `elemloc` architectural form, except that the `endterm` attribute's value must be the unique identifier

[9] HyTime semantics dictate that, in the absence of explicit `endterm` information, the anchor itself will somehow play that role.

of an element conforming to the Davenport "dav.choiceSignifier.HyBrid" architectural form (a validating Davenport application must check whether this is so), and a new attribute (colscope) has been added that provides a means of indicating the scope of the index collection (this, of course, is redundant information which may optionally be provided solely for convenience, since the scope is derivable by resolving the index element instance).

4.2.2 Glossary Set (dav.glossary-set.ilink)

```
<!-- Davenport Architectural Form.  Normative. -->
<!element dav.glossary-set.ilink    - O      (%HyBrid;) >
<!attlist dav.glossary-set.ilink
    Davnport CDATA    #FIXED     "dav.glossary-set.ilink"
    HyTime   NAME     #FIXED ilink
    id       ID       #IMPLIED
    reftype  CDATA    #FIXED
                            "linkends dav.dyad-link.ilink
                             endterm dav.choiceSignifier.HyBrid
                             endterms dav.choiceSignifier.HyBrid"
    -- The "dav.subject.HyBrid" end of the dav.dyad-link.ilink is the
       the term being glossed.  The "dav.predicate.HyBrid" end of the
       dav.dyad-link.ilink is the gloss. --
    endterm  IDREF    #IMPLIED
    linkends IDREFS   #IMPLIED
    extra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    intra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    endterms IDREFS   #IMPLIED
    aggtrav  -- lextype (("agg" | "mem" | "cor"),
                         (s+, ("agg" | "mem" | "cor"))*) --
             NAMES    agg
    context  (context | ncontext) ncontext >
```

The **glossary set** (dav.glossary-set.ilink) architectural form is for representing glossaries. It is built on the HyTime **independent link** (ilink) architectural form. It designates a collection of pairs of terms to be glossed and their respective glosses. Each pair is a dav.dyad-link.ilink form. The dav.subject.HyBrid end of the dav.dyad-link.ilink is the the term being glossed. The dav.predicate.HyBrid end of the dav.dyad-link.ilink is the gloss on the term specified by the dav.subject.HyBrid.[10].

[10]Multiple glossary sets can be melded together into a single glossary using an element conforming to the HyTime elemloc architectural form. If a single term appears in more than one glossary, however, and must appear only once

5 Support Declarations

The following module support declarations are required with the use of the HyTime architectural forms and options indicated in this DGDAS (see the HyTime IS for details).[11]

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
<?HyTime MODULE base exidrefs HyLex refctl >
<?HyTime MODULE measure >
<?HyTime MODULE locs
           aggloc anydtd anysgml coordloc notsrc pathloc query spanloc>
<?HyTime MODULE links manyends>
```

Since it relies on HyTime for its semantics, the DGDAS 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 DGDAS. However, future versions of the DGDAS that support specialized forms of HyTime activity tracking and/or tabular data representation via HyTime FCSs may require more elaborate arrangements. Therefore, the DGDAS uses the same powerful strategy for Davenport support declarations that is used in HyTime. This strategy gives future versions of the DGDAS the ability to add specialized architectural forms, while maintaining total backward compatibility and providing for future modularity:[12]

- A document indicates conformance to the DGDAS 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:

  ```
  Davenport=name
  ```

  if a prefix name is to be specified, otherwise "`Davenport`" 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.

in the compounded glossary, the application may have to detect and deal with that problem, even if the `elemloc`'s `set` attribute's value is `set`

[11] Additional HyTime support declarations may be needed when more HyTime architectural forms are incorporated by the DGDAS.

[12] These needs for future additions, backward compatibility, and modularity explain why there is, in the DGDAS 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. A Davenport support declaration is represented as an SGML processing instruction whose declaration name is "support." The name must be followed by one of the module names in the following list, thereby declaring that support is required for that module:

  ```
  base
  ```

  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,

  ```
  <?Davenport  VERSION "DGDAS:1992">
  <?Davenport  MODULE base  option1  option2>
  ```

  (There are currently no optional facilities in the DGDAS.)

The Davenport Base Module (`base`) requires support for all of the facilities described in this section and in Section 4 of the DGDAS.

6 Informative Annex: Sample Davenport Document

The following subsections contain a hypothetical DTD and document instance demonstrating the use of the DGDAS. The examples are not meant to be either comprehensive or profound, but merely illustrative. A diagram of the index and glossary structure of the sample instance is shown in Figure 1.

6.1 DTD for Sample Davenport Document

```
<?HyTime VERSION "ISO/IEC 10744:1992" HYQCNT=32>
<?HyTime MODULE base HyLex refctl >
<?HyTime MODULE measure >
<?HyTime MODULE locs aggloc coordloc query >
<?HyTime MODULE links manyends>

<?Davenport  VERSION "DGDAS:May 1992">
<?Davenport  MODULE base>

<!ENTITY % cntnt "section | HyLex | #PCDATA">
<!ENTITY % dyad  "glossub | gloss">
<!ENTITY % incl  "glosslnk | glossary | index | elemloc | strloc">
<!NOTATION grep SYSTEM "grep">
<!NOTATION ps   SYSTEM "ps">
```

```
<!ELEMENT sampledoc - O (%cntnt; | %dyad; | button | icon)*  +(%incl;)>
<!ATTLIST sampledoc
    Davnport  CDATA   #FIXED  "dav.document.HyDoc"
    HyTime    NAME    #FIXED  HyDoc
    id        ID      #IMPLIED >

<!ELEMENT section - O (para+) >
<!ATTLIST section id ID #IMPLIED >

<!ELEMENT para - O (partname | indexme | #PCDATA)+ >
<!ATTLIST para id ID #IMPLIED >

<!ELEMENT partname - - (#PCDATA) -- A name of a hardware component. -->
<!ATTLIST partname
    Davnport  CDATA   #FIXED   "dav.siteLocation.HyBrid"
    HyTime    NAME    #FIXED   HyBrid
    id        ID      #REQUIRED
    reftype   CDATA   #FIXED   "endterm button"
    endterm   IDREF   #IMPLIED >

<!ELEMENT indexme - - (#PCDATA)      -- An item to be indexed.  -->
<!ATTLIST indexme
    Davnport  CDATA   #FIXED   "dav.siteLocation.HyBrid"
    HyTime    NAME    #FIXED   HyBrid
    id        ID      #REQUIRED
    reftype   CDATA   #FIXED   "endterm button"
    endterm   IDREF   #IMPLIED >

<!ELEMENT button - O (%cntnt;) >
<!ATTLIST button
    Davnport  CDATA   #FIXED   "dav.choiceSignifier.HyBrid"
    id        ID      #REQUIRED
    conloc    IDREF   #CONREF
    butgraph  IDREF   #IMPLIED >

<!ELEMENT icon - O #PCDATA >
<!ATTLIST icon
    id        ID      #REQUIRED
    notation  NAME    #FIXED ps
    conloc    IDREF   #CONREF >
```

```
<!ELEMENT index  - O  (nmlist | nmidlst | nmquery)* >
<!ATTLIST index
    Davnport    CDATA   #FIXED     "dav.index.elemloc"
    collscope   IDREFS  #IMPLIED
    HyTime      NAME    #FIXED      elemloc
    id          ID      #REQUIRED
    ordering    (ordered | noorder)  noorder
    set         (set | notset)       set
    agglink     (agglink | nagglink) nagglink
    endterm     IDREF   #IMPLIED
    conloc      IDREF   #CONREF
    refrange    CDATA   #FIXED     "#ALL X endterm I"
    reflevel    CDATA   #FIXED     "#ALL -1"
    reftype     CDATA   #FIXED     "#ALL #ANY endterm button" >

<!ELEMENT gloss  - O  ((subview?,gloss) | %cntnt;)* >
<!ATTLIST gloss
    Davnport    CDATA   #FIXED     "dav.predicate.HyBrid"
    id          ID      #REQUIRED >

<!ELEMENT subview  - O (%cntnt;) >
<!ATTLIST subview
    Davnport    CDATA   #FIXED     "dav.subject-qualifier.HyBrid" >

<!ELEMENT glossub  - O  (glossub | %cntnt;)* >
<!ATTLIST glossub
    Davnport    CDATA   #FIXED     "dav.subject.HyBrid"
    conloc      IDREF   #CONREF >

<!ELEMENT glossend  - O  (glossub, gloss)>
<!ATTLIST glossend
    Davnport    CDATA   #FIXED     "dav.dyad">

<!ELEMENT glosslnk  - O  (#PCDATA) >
<!ATTLIST glosslnk
    Davnport CDATA    #FIXED  "dav.dyad-link.ilink"
    HyTime   NAME     #FIXED  ilink
    id       ID       #IMPLIED
    lexmodel NAMES    #FIXED  HyLex
    lextype  CDATA    #FIXED  "linkends idr2"
    reftype  CDATA    #FIXED  "linkends glossend #SEQ
                               endterm button
```

```
                              endterms button"
    endterm  IDREF    #IMPLIED
    linkends IDREFS   #IMPLIED
    extra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    intra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    endterms IDREFS   #IMPLIED
    aggtrav  -- lextype (("agg" | "mem" | "cor"),
                         (s+, ("agg" | "mem" | "cor"))*) --
             NAMES    agg
    context  (context | ncontext) ncontext >

<!element glossary  - O  (#PCDATA) >
<!attlist glossary
    Davnport CDATA    #FIXED "dav.glossary-set.ilink"
    HyTime   NAME     #FIXED ilink
    id       ID       #IMPLIED
    reftype  CDATA    #FIXED "linkends glosslink endterm button endterms button"
    endterm  IDREF    #IMPLIED
    linkends IDREFS   #IMPLIED
    extra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    intra    -- lextype(("E" | "I" | "A" | "N" | "P"),
                        (s+, ("E" | "I" | "A" | "N" | "P"))*) --
             NAMES    #IMPLIED
    endterms IDREFS   #IMPLIED
    aggtrav  -- lextype (("agg" | "mem" | "cor"),
                         (s+, ("agg" | "mem" | "cor"))*) --
             NAMES    agg
    context  (context | ncontext) ncontext >

<!NOTATION HyLex   PUBLIC
          "+//ISO/IEC 10744:1992//NOTATION HyTime Lexical Model Notation//EN" >
<!ELEMENT HyLex    -- HyTime lexical model --
                   - -   RCDATA >
<!ATTLIST HyLex    HyTime   NAME     #FIXED lexmodel
                   ltn      -- Lexical type name --
                            NAME     #REQUIRED
```

```
                  notation -- Data content notation --
                           NAME     #FIXED HyLex
                  tokens   -- Tokenization --
                           (tokens | notokens) notokens >

<!ELEMENT elemloc  - O      (nmlist | nmidlist | nmquery)* >
<!ATTLIST elemloc  HyTime   NAME     #FIXED elemloc
                   id       ID       #REQUIRED
                   ordering (ordered | noorder) noorder
                   set      (set | notset) set
                   agglink  (agglink | nagglink) nagglink >

<!ELEMENT nmlist   - O      (#PCDATA) -- lextype(NAMES) -->
<!ATTLIST nmlist HyTime   NAME     #FIXED nmlist >

<!ELEMENT nmidlist - O      (#PCDATA) -- lextype(NAMES) -->
<!ATTLIST nmidlist HyTime   NAME     #FIXED nmidlist >

<!ELEMENT nmquery  O O      (#PCDATA)* >
<!ATTLIST nmquery  HyTime   NAME     #FIXED nmquery
                   id       ID       #IMPLIED
                   notation NAME     #FIXED grep
                   qdomain  IDREF    #IMPLIED
                   qcontext IDREFS   #IMPLIED >

<!ELEMENT strloc   - O      (dimlist*) >
<!ATTLIST strloc   HyTime   NAME     #FIXED strloc
                   id       ID       #REQUIRED
                   overrun  (error | trunc | wrap) error
                   catsrc   (catsrc | nocatsrc) nocatsrc
                   catres   (catres | nocatres) nocatres
                   locsrc   IDREF   #IMPLIED  -- Default: previous specified --
                   ordering (ordered | noorder) noorder
                   set      (set | notset) set
                   agglink  (agglink | nagglink) nagglink >

<!ELEMENT dimlist  O O   (dimlist | dimspec | %marker; | #PCDATA)* >
<!ATTLIST dimlist  HyTime    NAME  #FIXED dimlist >

<!ELEMENT dimspec  - O   (marklist, marklist?) >
<!ATTLIST dimspec  HyTime   NAME     #FIXED dimspec
                   id       ID       #IMPLIED  -- Default: none -->
```

```
<!ENTITY % marker  "marklist | mrkquery">
<!ELEMENT marklist O O   (%marker; | #PCDATA)* >
<!ATTLIST marklist  HyTime   NAME     #FIXED marklist >

<!ELEMENT mrkquery O O      (#PCDATA)* >
<!ATTLIST mrkquery HyTime   NAME     #FIXED mrkquery
                   id       ID       #IMPLIED
                   notation NAME     #FIXED grep
                   qdomain  IDREF    #IMPLIED
                   qcontext IDREFS   #IMPLIED >
```

6.2 Sample Davenport Document Instance

```
<!DOCTYPE sample SYSTEM "sample.dtd"[
<!ENTITY lexdefs PUBLIC
           "ISO/IEC 10744:1992//TEXT Useful HyLex Lexical Types//EN">
]>

<sample>

&lexdefs;

<section id=sec1>

<para id=p1> When disassembling the <partname id=part1>instrument
bezel</partname>, be sure to save the <partname id=part2>grommet
</partname> and any <indexme id=ixme1>worn parts</indexme> for
later recycling.  </para>

<para id=p2> The importance of recycling <indexme id=ixme2>discarded
materials</indexme> cannot be overstressed for progress toward living
within our means.</para>

</section>

<button id=recyc1>recyclables</button>
<elemloc id=eloc1 endterm=recyc1><nmlist>ixme1 ixme2</elemloc>

<button id=cncpts>concepts</button>
<elemloc id=eloc2 endterm=cncpts><nmlist>p2 </elemloc>

<strloc id=str1 notation=grep locsrc=sec1>grep "recycling"</strloc>
```

```
<button id=recyc2 butgraph=ic1>recycling</button>
<icon id=ic1>. . . graphic info here . . .</icon>
<elemloc id=elocre endterm=recyc2><nmidlist>eloc1 eloc2 str1 </elemloc>

<button id=idxr>R</button>
<elemloc id=elocr endterm=idxr><nmidlist>elocre </elemloc>

<button id=idx>INDEX</button>
<index id=idx1 endterm=idx><nmidlist>elocr </index>

<button id=glss>Gloss</button>
<button id=subocc>Occurrence</button>

<glossub id=sub1 conloc=part1>
<gloss id=gloss1>
<subview>definition <gloss>An instrument bezel is a collar fitting used
to hold an instrument in place by pressure.</gloss>
<subview>Use<gloss>In emergencies, instrument bezels occasionally are
secured by shoelaces inserted through grommets in the bezel.</gloss></gloss>

<button id=glbut1 conloc=part1>
<glosslnk id=glnk1 endterm=glbut1 endterms=subocc glss linkends="sub1 gloss1">

<glossub id=sub2 conloc=part2>
<gloss id=gloss2>A grommet is a two-part metallic rim fitting used
to reinforce a circular hole.</gloss>

<button id=glbut2 conloc=part2>
<glosslnk id=glnk2 endterm=glbut2 endterms=subocc glss linkends="sub2 gloss2">

<glossub id=sub3 conloc=str1>
<gloss id=gloss3>Re-use of salvaged parts and materials</gloss>

<glosslnk id=glnk3 endterm=recyc2 endterms=subocc glss linkends="sub3 gloss3">

<button id=glsetbut>Glossary</button>
<glossary id=glset1 endterm=glsetbut linkends="glnk1 glnk2 glnk3">

</sample>
```

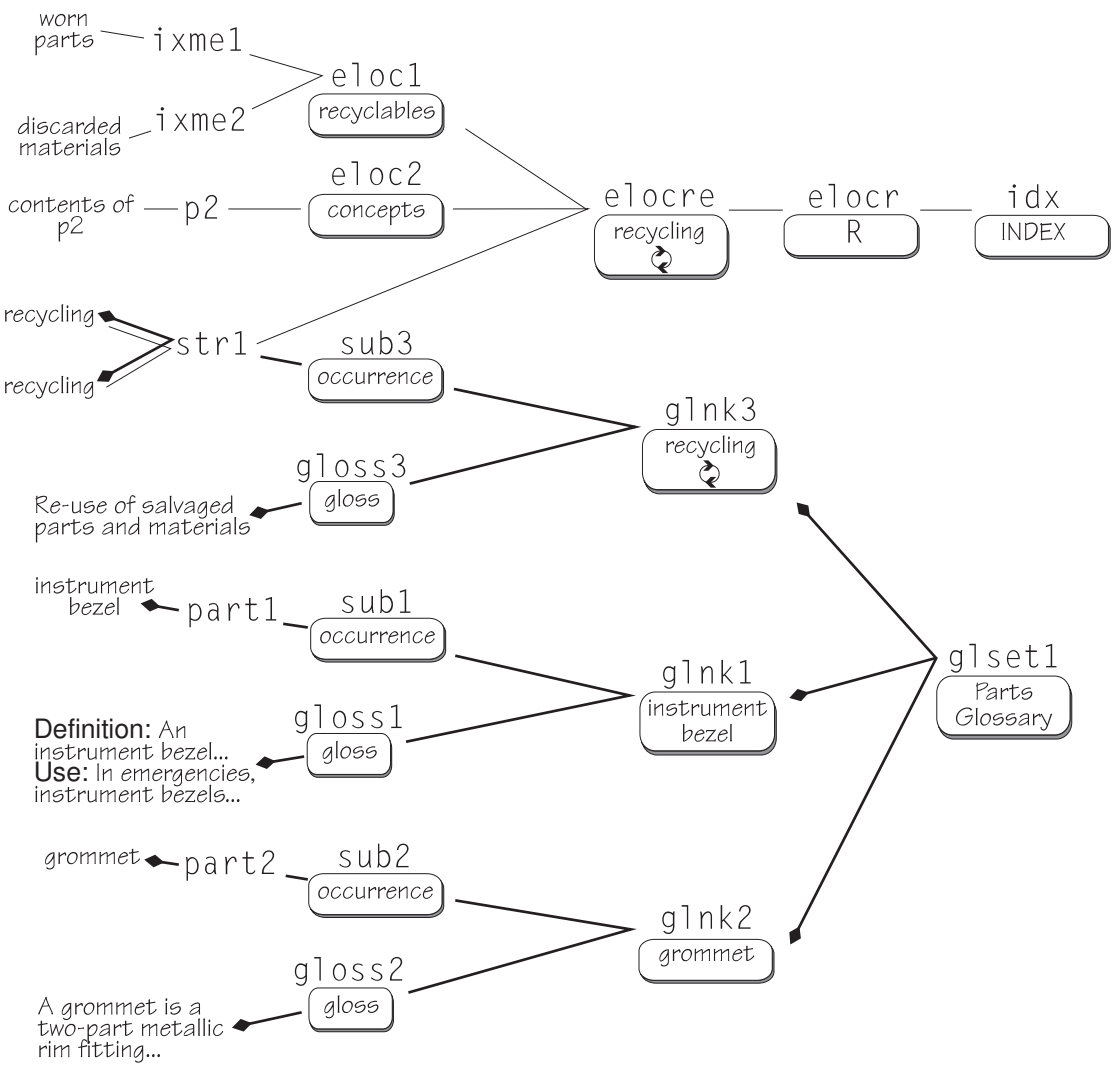

Figure 1: *Diagram of the index and glossary in a hypothetical Davenport-compliant document.*