Davenport Advisory Standard for Hypermedia (DASH) Module II: Conditions of Use for Commercial Hypermedia (COUCH) (Preliminary Straw Draft)

Editors: Steven R. and Victoria Taussig Newcomb*

DASH/COUCH 0.0.0, 11 February 1993

Contents

*This straw draft is made in preparation for the first COUCH Workshop, to be held immediately following the Graphic Communications Association's *TechDoc '93* Conference in San Antonio, Texas, on February 26 and 27, 1993. 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.

1 Introduction to the DASH and the COUCH.

This standard is a set of conventions for representing the wishes and policies of the owners and authors of DASH-compliant documents (including but not limited to hypermedia documents), regarding how the documents are to be used, and the terms on which various kinds of access to these documents are to be traded in the marketplace. This standard, called the "COUCH" (Conditions of Use for Commercial Hypermedia) standard, is one independently useful portion of the broader DASH (Davenport Advisory Standard for Hypermedia) standard.[1] [2]

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[3] 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 indepen-

[1] Thus far, the DASH has only one other module, the SOFABED (Standard Open Formal Architecture for Browsable Electronic Documents), which provides a platform-independent means of representing tables of contents, glossaries, indexes, and the knowledge bases that underlie them, as well as a means of establishing and expressing Davenport-defined and user-defined semantic roles for elements. See the SOFABED standard itself for further details.

[2] The SOFABED committee's report to the COUCH committee, which outlines the mission statement of the COUCH committee, forms Section 7 of this document.

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

dently of one another. The DASH will allow documents of dissimilar SGML document types[4] 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 same original document to be formatted for online delivery as hypermedia as well as for printing on paper.

In addition to the SOFABED and COUCH modules, one or more other DASH modules will standardize conventions for representing 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.[5] 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.

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.

[4]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.

[5]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.

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.[6] 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.[7]

1.2 The COUCH.

By contrast to the scope of the DASH, which is as yet not completely defined, the scope of the COUCH *is* well defined. The COUCH standardizes only the representation of "activity tracking" information – information having to do with the uses to which information may be or has been put.

The COUCH is intended, among other things,[8]

- to facilitate royalty accounting;

- to provide a standard protocol for notifying the authors of referencing documents when the referenced documents have been changed;

- to standardize the representation of an 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; and

[6]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 COUCH or SOFABED modules, 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.

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

[8]The SOFABED committee's report to the COUCH committee, which outlines the mission statement of the COUCH committee, forms Section 7 of this document.

- in general to allow universally understandable licensing and other legal information to accompany information objects wherever they may go and regardless of the circumstances under which they are ultimately made available to end users.

2 References and Resources.

1. ISO/IEC[9] 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

[9] "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 COUCH 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.[10]

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.

[10] 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.[11]

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[12] and/or by added HyTime `lextype` and `reftype` attributes.

HyTime "common attributes"[13] 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.[14]

[11] 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.

[12] *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.

[13] 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.

[14] 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,[15] 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 standard).

4 COUCH Architectural Forms.

> *****EDITORS' NOTE***** For this straw draft, this (list of only one) form is merely what was excised from the SOFABED and referred to the COUCH committee for consideration. It has not been edited, and its appearance herein does not imply anything about the architectural forms that will appear in future drafts (much less the final form) of the COUCH.

4.1 Activity policies (`dav.policies`).

Activity policies (`dav.policies`) is merely a template for dummy element declarations which are intended to allow various `reftype` attributes to restrict the element types to which `activity` attributes appearing in the same attribute list may refer.[16] In an actual DTD, the situation may not require such a dummy element, or it may require several such dummy elements.

```
<!element dav.policies  - O (%HyBrid;)* >
```

- 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.

[15] 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.

[16] See the HyTime IS for details about the `reftype` and `activity` attributes.

```
    <!-- Template for dummy resource elements that serve
         to specify the element types that activity attributes
         may refer to. -->
<!attlist dav.policies
    Davnport    CDATA    #FIXED        dav.policies
>
```

5 HyTime Support.

Assuming that no other DASH module is supported, the minimum required HyTime support is declared as follows:

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

6 Davenport Support.

Since it relies on HyTime for its semantics, the COUCH 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 COUCH or SOFABED. However, future versions of the DASH that support, e.g., tabular data representation via HyTime FCSs, may require more elaborate arrangements. Therefore, the DASH 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.[17]

A document indicates conformance to the COUCH 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.

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

A DASH 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,[18] thereby declaring that support is required for that module:

`couch`

`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 couch  option1  option2>
```

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

The Davenport COUCH Module (`couch`) requires support of the HyTime standard at the level(s) described in Section 5, as well as support of the facilities described in Section 4 of this Standard.

7 Preview of Activity Tracking and Management in the COUCH Standard.

> *****EDITORS' NOTE***** This is the mission statement of the COUCH committee as developed by the SOFABED committee. In all likelihood, the substance of this statement will be redistributed throughout the COUCH standard in future drafts.

There is need for a set of clear, concise, and unambiguous statements regarding the owners' and/or authors' intentions with respect to the uses to which the information contained in the document may be put. The laws of most countries provide, for example, ways to inform users that certain material is copyrighted, and that certain rights are reserved unto the owners. Another familiar example, at least in the military milieu, is the secrecy-level classification system.

In the emerging world of electronic publishing, many comparatively new kinds of access are possible; electronic information may be conveniently used in many more ways than information printed on paper. Information can be bought and sold online without ever being put into some

[18] Because the SOFABED and COUCH are currently the only modules of the DASH, there are only two items in the list at the time of this writing.

tangible form. The publishing industry is becoming far more complex, as qualitatively different kinds of access to the same information (e.g., query access, full text access, access with right to link to, access with ability to contribute to, access with time limit, access to some version, etc.) are being provided and marketed in an exploding number of ways. The right of an author to include in his work a traversable link to information owned by another author is particularly relevant to the interests of the Davenport Group. Such links are a hallmark of electronic hypertext publishing, and and their growing importance demands a detailed rethinking of the entire publishing industry. The Davenport Group takes the view that the publishing industry is no longer best understood as an industry whose primary focus is the promotion, manufacture, and distribution of tangible goods. Instead, the electronic publishing industry's primary purpose is to distribute rights in information; there need not be any tangible product at all, or, if there is a tangible product, the most important aspect of purchasing such a product is not the tangible product itself, but its associated license to use it – i.e., the uses to which that product may be put by its purchasers.

It is beyond the scope of the Davenport Group's activities to develop and standardize specific packages of rights. Similarly, it is not the intention of the Davenport Group to reinvent the publishing industry or even to make guesses about the types of businesses that will come into being. The purpose of this Annex's suggested elaborations of HyTime's "activity tracking" facilities is merely to provide a base on which the widest possible variety of information products, rights packages, and old and new business models can be built. These "activity tracking" facilities are mechanisms for expressing policies; they are not intended to set forth, or even to suggest, the policies themselves.

7.1 Categorizing Activity Tracking and Management Policies.

For the purposes of this discussion, "activity tracking policies" are broken into two categories, owner's policies and author's policies.

7.1.1 Owner's Law.

An owner's activity tracking policy has the force of law. An owner can charge admission, discriminate between users, require the concurrent display of a logo, demand detailed reports on the usage habits of consumers, order the destruction of the information under certain conditions, etc.

Permissions. Only an owner of a document can grant permissions to access, query, quote, modify, destroy, annotate, link, unlink, copy, resell, etc. etc.[19] Under this rubric can come owner requirements relating to subscriptions, billing, usage measurement (metered by time, number of characters, number of machine cycles, number of links traversed, number of elements delivered, etc.), authorization, and authentication.

Essential to the idea of brokering access is the idea of the *offer.* An offer is a hypothetical *quid pro quo* – a set of prerequisites, in exchange for which certain rights will be granted.

[19] How the owner's permissions are actually brokered and policed is not at issue. The COUCH module is concerned only with how to represent the intentions of owners and authors with respect to their information. The brokering of permissions can be accomplished in any number of ways, from highly secure sales and delivery systems to shareware houses operating on an honor system.

The **prerequisites**, or conditions, of an offer may involve:

1. **price**, terms of payment, issuance of credit, promotional programs and privileged pricing, and taxes;
2. **security clearance** and membership in some class of privileged users;
3. **registration** as a licensed user;
4. **exposure** to warning notices, advertisements, etc.;
5. **agreement** to a wide variety of other conditions, waivers of rights, etc.; and
6. **acceptance** of some previous offer.[20]

The **rights** granted if the purchaser meets the conditions of the offer may involve:

1. The right to **modify**, edit, abridge, censor, or augment;
2. The right to **copy** electronically (i.e., copy source and/or a formatted instance), make an electronic copy for backup purposes only, make a printed copy, reprint for others, quote briefly in critical reviews, etc., or quote extensively;
3. The right to **annotate**;
4. The right to **publish**, buy all rights, lease, include in an anthology, include in the bounded object set (BOS)[21] of a hyperdocument, use in an unattributed fashion, reprint for sale, and publish with annotations, augmentation, diminutions, variations, etc.;
5. The right to **translate**, transcribe, transform for delivery via a particular medium, etc.;
6. The right to **browse** freely, browse with content-based billing, browse with time-based billing, execute, run, play, or perform in private or in public, etc.;
7. The right to **preview**, access a summary, abstract, or outline, get a guided tour, see the table of contents, or see a description as in a catalog, etc.;
8. The right to **search and/or index**;
9. The right to **access the bibliographic information**, i.e., the edition(s), author(s), title(s), owner(s), etc.; and
10. The right to **access the source code**, i.e., to see and/or otherwise use the markup.

[20] An additional complication is the fact that certain offers may only be made after other, prerequisite offers have already been accepted. This complication doesn't change the basic idea of an offer as a *quid pro quo*, though.

[21] See HyTime standard.

Effectivity. An owner must be able to make unambiguous statements regarding the situations to which any particular datum is relevant, thus to prevent, if the owner requires it, the appearance of that datum except when it is intended to be effective.

Instrumentation and reporting. An owner may demand detailed reports regarding who accessed which information, in order to support marketing and product development decisions. For example, if the owner's policy is to offer some kind of access in exchange for some fee, how many users decline the offer, and by what path did the various decliners arrive at the offer? Under this rubric can come owner requirements regarding usage measurement and tracking.

7.1.2 Author's suggestions and advice.

An author's activity tracking policies or advice concerning the use of his work may be taken as suggestions. An author may, for example, create certain portions of a document that are only intended to be used by certain users, or under certain circumstances. Some examples:

1. **Suggestions based on user preferences.** For example, details may be skipped over (i.e., not shown) if the user's intent is to get the general idea, but they may appear if the document is being used as a reference work. The author may choose to support such skimming by indicating (via activity tracking policies that govern access) which points are most important.

2. **Suggestions based on user's previous experience.** Different paths through a document may be chosen on the basis of what has been done by a user during the current and/or previous sessions. For example, if, at a given menu, one choice has already been made, an author may arrange the document in such a way as to make that choice unavailable the second time around. An author may choose to record in each user's knapsack[22] the fact that the user has seen a given portion of a document, and to use those records in making suggestions to a user as to which things the user should see next.

3. **Suggestions based on the relevance and/or effectivity of the information.** For example, instructions as to how to invoke a program might be different depending on whether the user is currently using one command-line interface (shell) rather than another (e.g., csh rather than bash). Similarly, maintenance instructions for a particular model of aircraft engine might be hidden from a maintenance worker who is working on a different model.

4. **Instrumentation and reporting.** An author may request a variety of reports intended to reveal how and why users of the information are using it. For example, if the author's suggestion is to skip the first chapter if the user already knows the basic information, how many users chose to use the first chapter and how many skipped over it.

   An author may also choose to provide means whereby a user can report errors, omissions, and other problems directly to the author (or current maintainer) of the information, along with an automatically generated indication of where in the information the commenter was when the comment was written.

[22] See Section 7.2.3

Of course, in many cases it is likely that users may wish to override an author's built-in guidance mechanism, and it will be reasonable in most cases for delivery applications to allow such overriding to occur. For example, a person who is using the csh shell may still want to know how to invoke the program under the bash shell.

The ability for a user to override an author's suggestion contrasts sharply with a user's inability to circumvent an owner's demand for payment (see Section 7.1.1, above) as a minimum requirement for access. In general, there appears to be a kind of continuum of the amount of user discretion in the various kinds of things that should be representable in Davenport activity management elements:

← less user discretion		*more user discretion →*	
permissions, rights, prices, etc.	required experiences, qualifications & conditions	relevance and effectivity	user preferences

7.2 Support for Activity Tracking and Management.

In order to support all these functions, at least three new Davenport-specific facilities are needed:

1. A means of encapsulation (i.e., an architectural form) for expressing specific intentions regarding the use of information. The primitive HyTime "activity" architectural form set is used as a basis for the Davenport `dav.activityManager.activity` form. HyTime's "activity" common attribute is used to associate `dav.activityManager.activity` elements with all other elements. The `actypes` attribute defined by HyTime provides additional classifying information.[23]

2. Some kind of script language capable of expressing owners' and authors' intentions with respect to the use(s) to which information may be put.[24]

3. A convention for expressing interchangeable user-specific information – a "knapsack" document type that would be used to collect user history.[25]

What follows is merely a preliminary design survey of the three above-outlined facilities needed for effective and standardized expression of owners' and authors' intentions.

7.2.1 Davenport Activity Manager.

In a Davenport document, all elements that conform to the HyTime `activity` architectural form must also conform to the **activity manager** (`dav.activityManager.activity`) form. Activities with respect to all aspects of any element whose HyTime `activity` common attribute points to

[23]See Section 6.5.7 of the HyTime standard, and Section 7.2.1 of this standard.

[24]See Section 7.2.2.

[25]See Section 7.2.3.

an element conforming to the Davenport architectural form `dav.activityManager.activity` are governed by the policy(ies) set forth in that `activityManager` element. If an element contained in another element points to an `activityManager` element that expresses a policy in conflict with the policy pointed to by the containing element, the conflicting policy pointed to by the contained element overrules the containing element's `activityManager` element's policy. All non-conflicting policies of all elements and of all elements that contain them are applied; the order in which they are applied is not defined herein.

```
<!element dav.activityManager.activity
          -- The expression of the owner's and/or author's intent with
             respect to the use of the information.  --
          -- Content is DavAct script language program; it may be a
             subroutine returning data.--
                             - O (#PCDATA) >
<!attlist dav.activityManager.activity
                Davnport   NAME    #FIXED    dav.activityManager.activity
                HyTime        NAME    #FIXED    activity
                id         ID      #REQUIRED >
```

7.2.2 Script Language.

The **"DavAct" Script Language** is not yet designed. Among the desiderata and possible requirements to be satisfied by DavAct's design are:

1. If/then/else facilities.

2. Expressions that can return true or false. It might be a good idea to let them return enumeration types or even data as well.

3. Ability to embed HyQ and HyFunk statements. In general, it is desirable to use existing HyTime constructs and techniques wherever appropriate and possible, as this will facilitate understanding by users and implementation by SGML/HyTime applications builders.

4. Ability to create, destroy, and assign values to variables in dynamic "knapsack" documents, and, of course, to determine the existence and value of variables in a knapsack. (Also, the ability to create, read, and write specific "knapsack" documents.)

5. Ability to contain SGML unique element identifiers. These could point to HyTime location address elements, and thereby to anything at all. One important purpose of referring to other local elements would be to use other `dav.activityManager.activity` elements as subroutines.

6. Ability to be learned and promulgated easily. It should be based on some well-regarded and widely used existing convention, such as Lisp, Smalltalk, C++, Scheme, etc. etc.

Here is a pseudo-example of a DavAct script. (Obviously, in the absence of a design for DavAct, there is no way to give an example. The purpose of this example is merely to give an idea of the kinds of things that DavAct is supposed to represent, and to demonstrate the inclusion of some relevant conventions drawn from HyQ and HyFunk.)

```
<permit.read Davnport=dav.activityManager.activity>
            HyTime=activity
            id=rp1231
            actypes="access">
    if (
        AND(EQ(u.security.level, ‘‘TOPSECRET’’), GE(u.money.acctval, 10))
                        -- if user’s security level is "TOPSECRET"
                           (as given in user’s "security" knapsack) and
                           user has 10 dollars (?) available in his "money"
                           knapsack --
    ) then (
        ASSIGN(u.money.acctval, @sub(u.money.acctval, 10))
                                -- take 10 from user --
        ASSIGN(o.money.acctval, @add(o.money.acctval,  8))
                                -- give 8 to owner(s) (o. ...) --
        ASSIGN(a.money.acctval, @add(a.money.acctval,  2))
                                 -- give 2 to author(s) (a. ...) --
        ASSIGN(u.license, "read timelimit", "24 hours", NOW)
                                 -- give user a 24 hour license to read.
                                    License is stored in the variable
                                    "u.license.rp1231". --
        ASSIGN(u.permit.rp1231, "TRUE")
                              -- Give user permission to do whatever
                                 this dav.activityManager.activity
                                 controls. --
        RETURN(TRUE)          -- Inform application it’s ok to proceed. --
    ) else (
        if (
            AND(EQ(u.security.level, ‘‘TOPSECRET’’), VALID(u.license.rp1231))
                        -- if user’s security level is "TOPSECRET"
                           (as given in user’s "security" knapsack) and
                           user has a valid license to use --
        ) then (
            RETURN(TRUE)
        ) else (
            RETURN(FALSE)
        )
</permitread>
```

7.2.3 Knapsack Document Type.

The purpose of the **knapsack** document type provides a convention for representing interchangeable knapsack data. A DTD for knapsack documents needs to be designed, and a short list of knapsacks that may be associated with users and documents should be defined. The below list is merely a starting point for discussion:

Knapsacks carried by users may contain:

1. Information about the user's working environment, e.g., GUI/windowing system, type of shell, condition of environment variables, equipment being used, available network connections, etc.

2. Information about the user as a person, e.g., work experience, membership in professional associations, security clearances, personal preferences, etc.

3. Information about the user's current session, e.g., how (via what link, etc.) the user arrived at this document, which parts of this document has the user already seen, etc.

4. Information about the user's previous exposure to this document.

5. Information about a variety of preferences the user has, e.g., which GUI is the user most comfortable with, what things has the user set default values for, etc.

Knapsacks carried by documents may contain:

1. Information about where users came from, i.e., what links were traversed in order to arrive here.

2. Information about the path users took when browsing in this document.

3. Information about ongoing editing, annotation, and linking activities involving this document.