1 Activity Tracking and Management

[Editors' Note: All of the below is an expression of the Editors' understanding of the issues brought out in the course of the initial three days of discussion of this topic at the DASH committee meeting held August 16-18 in Sebastopol, CA. This section has not been reviewed by the committee, and, inasmuch as it differs in several respects from the tentative conclusions drawn at those initial discussions, it can not (at this writing, at least) be regarded as any sort of statement of the committee's position. In particular, the committee tentatively concluded that an activityManager element might have an `op (record|erase|query) #REQUIRED` attribute and a `class (permission|experience|relevance|preference) #REQUIRED` attribute. When the Editors considered how to combine activityManager elements in late-binding conditional statements (and saw that such complex statements demanded some sort of script language anyway), it seemed unduly arbitrary and clumsy to put the `op` and `class` attribute semantics outside the script language. Obviously, all decisions made by the Editors are subject to reversal by the committee.]

There is need for a set of clear, concise, and unambiguous statements regarding the owners' and/or authors' intentions with respect to the uses, modes of use, etc. 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 to 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 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 [or version set], 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. This DASH 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 this DASH to develop and standardize specific packages of rights. Similarly, it is not the intention of this DASH to reinvent the publishing industry or even to make guesses about the types of businesses that will be invented or reinvented. The purpose of this DASH'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.

This DASH's definition[1] of "Activity Tracking Policy" is: '

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.

   (a) **Permissions of various kinds.** Only an owner of a document can grant permissions to access, query, quote, modify, destroy, annotate, link, unlink, copy, resell, etc. etc.[2] 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.[3]

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

         A. **price**, terms of payment, issuance of credit, promotional programs and privileged pricing, and taxes;

         B. **security clearance** and membership in some class of privileged users;

         C. **registration** as a licensed user;

         D. **exposure** to warning notices, advertisements, etc.

         E. **agreement** to a wide variety of other conditions, waivers of rights, etc.

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

         A. The right to **modify**, edit, abridge, censor, or augment;

         B. 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 *in extenso*;

         C. The right to **annotate**;

         D. The right to **publish**, buy all rights, lease, include in an anthology, include in the bounded object set (BOS) of a hyperdocument, use in an unattributed fashion, reprint for sale, and publish with annotations, augmentation, diminutions, variations, etc.;

         E. The right to **translate**, transcribe, format for a particular medium, etc.;

         F. The right to **browse** *ad libitum*, browse with content-based billing, browse with time-based billing, execute, run, play, or perform in private or in public, etc.;

[1] Thanks to Dave Hollander.

[2] How the owner's permissions are actually brokered and policed is not at issue. This DASH 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.

[3] 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.

G. 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.;

H. The right to **search and/or index**;

I. The right to **access the bibliographic information**, i.e., the edition(s), author(s), title(s), owner(s), etc.; and

J. The right to **access the source code**, i.e., to see the markup.

(b) **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.

(c) **Required experiences and qualifications of users.** A very common example of this kind of thing is the requirement that certain safety warning notices be read and understood, and perhaps even disclaimers acknowledged and agreed to by users. Similarly, and for a wide variety of reasons, information owners may require users to maintain membership in some professional organization, or to have some security clearance, or to have other qualifications such as certain academic degrees in order to gain access. Under this rubric can come owner requirements regarding notification, accreditation, authorization, authentication, and registration.

(d) **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.

2. **Author's suggestions and advice.** 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:

(a) **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.

(b) **Suggestions based on user's previous experience.** Different paths through a document may be chosen on the basis of the tokens found in a user's appropriate "knapsack," – the record 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 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.

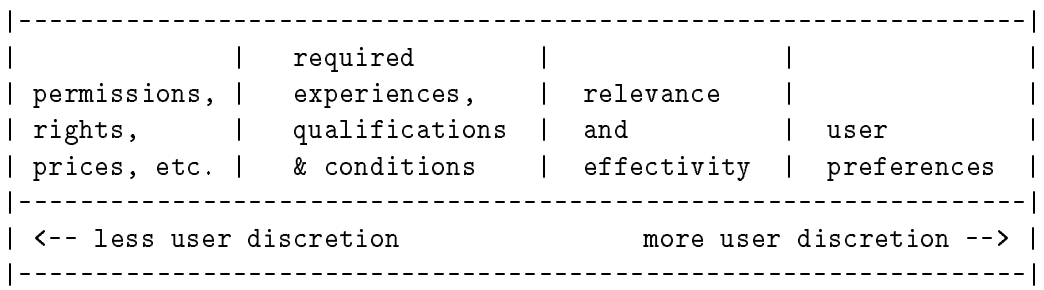
(c) **Suggestions based on the relevance and/or effectivity of the information.** For example, instructions as to how to invoke a program might be different depending on whether the user is currently using one command-line interface (shell) rather than another (e.g., csh vs. bash). Similarly, maintenance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

(d) **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, lacunae, 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.

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 (see item 2, above) contrasts sharply with a user's inability to circumvent an owner's demand for payment (see item 1, above) as a *sine qua non* 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 `activityManager` elements:



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. This DASH elaborates the HyTime `activity` form by providing the classifying attributes `op` and `class`. The `actypes` attribute defined by HyTime provides additional classifying information.[4]

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.

3. A convention for expressing interchangeable user-specific information – a "knapsack" document type.

What follows is merely a first (and only partial) attempt to design the above-outlined facilities needed for effective and standardized expression of owners' and authors' intentions.

1. **dav.activityManager.activity** ("actman"). In a Davenport document, all elements that conform to the HyTime `activity` architectural form must also conform to the dav.activityManager.activity form. Activities with respect to all aspects of any element whose HyTime `activity` common attribute points to some actman element (i.e., to some element conforming to the Davenport architectural form `dav.activityManager.activity`) are governed by the policy(ies) set forth in that actman element. If an element contained in another element points to an actman that expresses a policy that conflicts with the policy pointed to by the containing element, the conflicting policy pointed to by the contained element overrules the containing element's actman'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 by this DASH.

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

<!attlist dav.activityManager.activity
Davnport NAME #FIXED dav.activityManager.activity
HyTime  NAME #FIXED activity
id ID #REQUIRED>
```

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

   (a) If/then/else facilities.

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

[4] See ISO/IEC 10744:1992(E) pp. 32-34.

(c) 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 HyTime engine builders.

(d) 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.)

(e) Ability to contain unique local element identifiers. These could point to location address elements, and thereby to anything at all. One important purpose of referring to other local elements would be to use other actmans as subroutines.

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 license to read.
                                license is stored in the variable
                                "u.license.rp1231". --
        ASSIGN(u.permit.rp1231, "TRUE")
                                -- give user permission to do whatever
                                   this actman 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>
```

3. 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:

   (a) **Knapsacks carried by users may contain:**

      i. 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.

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

      iii. 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.

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

      v. 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.

   (b) **Knapsacks carried by documents.**

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

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

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