Handheld Device Markup Language FAQ

What is HDML?

The Handheld Device Markup Language (HDML) is a simple language used to define hypertext-like content and applications for hand-held devices with small displays. HDML is designed to leverage the infrastructure and protocols of the World Wide Web while providing an efficient markup language for wireless and other handheld devices. Congruent with the capabilities and limitations of many handheld devices, HDML's focus goes beyond presentation and layout. HDML provides an explicit navigation model which does not rely upon the visual context required of HTML. As such, HDML offers an efficient means of providing content via the WWW infrastructure to handheld devices such as cellular phones, pagers, and wireless PDA's.

Is HDML designed to allow general web access from devices with small displays?

No. The vast majority of content on the web would not be useful on such a constrained device. The devices for which HDML is a target have very small displays (e.g., 4 lines by 20 characters), limited compute resources (e.g., RAM, CPU, etc.), and low bandwidth (e.g., 9600 bps or less). Examples include mobile phones and two-way pagers.

The goal is not to display any given HTML page on the device, but rather to use the vast infrastructure of the web to provide data and content in a way which is easy to use and efficient concerning the constrained resources of the device.

Why not use HTML or a subset of HTML?

HTML requires significant context (visual or otherwise) to be useful. The constrained display and restricted resources of handheld devices reduce the context to such a level that HTML becomes unusable.

The suggestion is made to use only a subset of HTML and design pages specifically for small displays. However, the navigation model for HTML still requires context not available on a small-screen device. An explicit navigation model is required to make data access efficient and useful on these devices.

What is the "explicit navigation model" of HDML?

The navigation model in HDML is optimized for super-small displays (e.g. 4x20 characters) which do not have a lot of screen space to provide context for the user in the form of "Prev" and "Next" buttons or a pull-down menu of the history. To help prevent the user from "getting lost", the HDML navigation model emphasizes predictability, and determinism by providing explicit control over the navigational UI of the user agent.

The HDML user-interface metaphor is based on "cards" that the user displays and/or interacts with. The navigation model is similar to the HTML model in that there is a notion of moving "forward" and "back". "Forward" usually means display the next card, and "back" usually means show me the previous card.

When building an application interface using small cards, a number of problems arise using a simple history model. For example, if the URL of a card has a side-effect when downloaded from a server, it could be a bad thing for the user to move back to the previous card and then forward again, reinvoking the side-effect. As another example, it is not possible to collect input with a series of cards and then be done with them, i.e. remove them from the history, to bring the user back to the card he or she started from.

The HDML language provides explicit control over both the history and the navigational UI. An HDML card can explicitly define the action to take when certain keys are pressed. Also, cards can be grouped into "activities" so they can be removed from the history in a single operation.

What are variables in the sense of HDML?

Variables are client data that can be substituted directly into HDML by the user agent. Variables can be substituted into both the content (i.e. the text) and the markup parts of HDML. Variables are useful for the following:

- Parameterizing cards
- Sharing data between cards

Parameterized cards are useful when you have a number of cards that are the same except for a few minor differences. Let's say, for example, you have built an application which displays email messages, and each message can bring up a sub-menu of operations (e.g. reply, forward, delete). For each message, the sub-menu is the same except for the identity of the message to operate on. With HDML variables, the message identity can be stored in a variable allowing all messages to share the same sub-menu of operations. This results in both a dramatic savings of cache space in the user agent and a reduction of network bandwidth.

Variables also allow data to be shared between cards. This provides a means by which the user can navigate through a series of cards to build up a data set that is sent to the server in one final request. This is not unlike HTML forms, except the fields are spread across several cards, and the values can be used in a wider variety of ways. HDML also provides a means to "return" variables from cards. One application of this we have found particularly useful is a centralized preferences service. The preference service provides a cached set of cards which return the preferences settings in variables. This allows applications to utilize centralized preferences without incurring network overhead.

The variables support in HDML is not "scripting" per se. There is no control-flow operations, branching statements, or the like. Variables are set via navigational operations and user input. The variable support is a lightweight mechanism for client-side data sharing.

Why not use XML?

The genesis and evolution of HDML has occurred over the past two years. XML has only recently become available and further study is required. However, despite the name, HDML is more than a markup language. Some of the features and functions embodied in the language -- and which make it so well suited for small-display devices -- may not be possible in XML. If XML is found capable of supporting the features and functions of HDML which allow efficient use on constrained devices, then HDML should be recast as XML.

What about Cascading Style Sheets?

CSS defines how the contents of an HTML page should be displayed or rendered. Defining a style sheet to display data that looks good on a small-screen devices, however, would not solve the problem of navigation vagaries found when HTML is used on context-constrained devices.

Will people have to convert their HTML pages to HDML?

Most HTML content makes no sense on a small device. Those pages with time-sensitive, discrete pieces of data do make sense, however. Canonical examples include stock quotes, weather reports, e-mail, calendar and appointment data, yellow pages and white pages. Intranet data as well: inventory data, price lists, corporate directories, invoice status. Additionally, using the interactive nature of the web and the interconnectivity of databases, applications available to wireless hand-held devices can initiate actions to be completed by a server: fax a document, submit a purchase order, etc.

Fortunately, most data of this class is not stored in the form of static HTML pages. Rather, the data is stored in databases or accessed via some form of telemetry. In today's web, Perl scripts and query tools retrieve data based on incoming URLs and return data formatted for display on an HTML user agent. Those sames tools can easily be modified to return data formatted in HDML instead.

A technology developed solely for data access on cell phones and pagers seems like a niche technology. What else is HDML useful for?

While the target design point for HDML was wireless devices with small displays and constrained resources, HDML will also be useful in other environments where visual context is limited and where there is a need for concise and explicit transactions. While HDML can't provide (and isn't meant to provide) access to the body of content found on the web for the sight-impaired, it may provide an alternate mechanism for accessing some classes of important data.

Who uses HDML today?

HDML first became available to the public in mid 1996. Since then, almost 2000 developers have downloaded an HDML application development kit from Unwired Planet's web site. Over 2300 HDML applications have been registered with Unwired Planet, as of April 1997.

Commercial content providers who provide data access via HDML include Infospace (directories), Data Broadcasting Corp (financial news and stock quotes), FedEx and UPS (package tracking), and the

Weather Underground (weather reports), among many others.

Mobile phone service providers in the US who offer HDML services include AT&T, Bell Atlantic NYNEX Mobile, Ameritech, and GTE Wireless.

Manufacturers of mobile phones who provide phones with built-in HDML user agents today or plan to offer phones in the near future include Alcatel, Mitsubishi, Motorola, Qualcomm, Samsung, Uniden, and PCSI. Vendors of operating systems used in mobile phones also support HDML: Microware and Geoworks.

Vendors of web development tools which support HDML include Allaire (Cold Fusion), Oracle (Oracle Web Server), the Sapphire Group (Page Blazer), and Universal Access (Meta-HTML).

HDML 2.0 FAQ - 11 APR 1997
For more information about HDML, contact:
Tim Hyland
Product Manager
Unwired Planet