

Network Working Group  
Internet-Draft  
Expires: March 7, 2006

R. Sayre  
September 3, 2005

XHTML Microformats for the Atom Publishing Protocol  
draft-sayre-atompub-xhtml-micro-00.txt

## Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on March 7, 2006.

## Copyright Notice

Copyright (C) The Internet Society (2005).

## Abstract

This memo presents a number of XHTML microformats for use with the Atom Publishing Protocol.

## Editorial Note

To provide feedback on this Internet-Draft, join the atom-protocol mailing list (<http://www.imc.org/atom-protocol/index.html>) [[1](#)].

Internet-Draft

XHTML Microformats for the APP

September 2005

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Notational Conventions . . . . .	<a href="#">4</a>
<a href="#">3.</a>	hCat: Atom Categories . . . . .	<a href="#">5</a>
<a href="#">3.1</a>	Creating and Editing hCat Categories . . . . .	<a href="#">7</a>
<a href="#">4.</a>	hError: Error Documents . . . . .	<a href="#">9</a>
<a href="#">5.</a>	Security Considerations . . . . .	<a href="#">11</a>
<a href="#">6.</a>	References . . . . .	<a href="#">12</a>
<a href="#">6.1</a>	Normative References . . . . .	<a href="#">12</a>
<a href="#">6.2</a>	Informative References . . . . .	<a href="#">12</a>
	Author's Address . . . . .	<a href="#">13</a>
<a href="#">A.</a>	An Example hCat and hError Server . . . . .	<a href="#">14</a>
<a href="#">B.</a>	An Example hCat and hError Client . . . . .	<a href="#">20</a>
	Intellectual Property and Copyright Statements . . . . .	<a href="#">31</a>

---

Internet-Draft

XHTML Microformats for the APP

September 2005

## 1. Introduction

Atom Publishing Protocol [[APP](#)] client implementations require a fair amount of ancillary server-provided data in order to provide a smooth user experience. Rather than invent a plethora of new XML formats, this specification chooses to present a number of XHTML profiles [[XHTML](#)], colloquially known as "microformats". Visit <http://microformats.org> for more information.

## 2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

### [3.](#) hCat: Atom Categories

hCat is an XHTML profile for encoding the three standard attributes of Atom category elements [[AtomFormat](#)]. By providing a definition list containing encoded category information, servers can present clients with a list of known categories in an XHTML definition list. hCat also allows description of endpoints for category editing through a simple HTTP-based protocol, described in [Section 3.1](#).

A sample category definition

```
...
<dt>birds</dt>
<dd>
  <span class="label">ISBN birds</span>
  <a href="http://example.org/category/4" rel="hCat">Edit</a>
  <a href="http://example.org/ISBN/" rel="scheme"></a>
</dd>
...
```

The three standard properties of an Atom category are 'term', 'label', and 'scheme'. The 'term' attribute is required. hCat encodes the categories provided by a server within an `xhtml:dl`

element with a 'class' attribute value of 'category'. Each category is presented as a pair of elements: xhtml:dt and xhtml:dd. Each element pair MAY contain or be enclosed by additional markup, but this specification assigns it no significance.

hCat XHTML documents MUST indicate their profile in the 'head' element. The profile URI for hCat documents is "http://www.example.org/2005/Atom/hCat" [example URI--do not deploy].

An example hCat document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://www.example.org/2005/Atom/hCat">
    <title>Atom Category Demo</title>
  </head>
  <body>
    <h1>Atom Category Demo</h1>

    <form class="hCat" action="http://example.org/cat" method="post">
      <input id="term" type="text" />
      <input id="label" type="text" />
      <input id="scheme" type="text" />
    </form>
```

```

<dl class="hCat">

  <dt>cats</dt>
  <dd>
    <a href="http://example.org/category/1" rel="hCat">Edit</a>
  </dd>

  <dt>dogs</dt>
  <dd>
    <a href="http://example.org/category/2" rel="hCat">Edit</a>
    <a href="http://www.example.com" rel="scheme"></a>
  </dd>

  <dt>atom</dt>
  <dd>
    <span class="label">The Atom Syndication Format</span>
    <a href="http://example.org/category/3" rel="hCat">Edit</a>
  </dd>

  <dt>birds</dt>
  <dd>
    <span class="label">ISBN birds</span>
    <a href="http://example.org/category/4" rel="hCat">Edit</a>
    <a href="http://example.org/ISBN/" rel="scheme"></a>
  </dd>

</dl>
</body>
</html>

```

The REQUIRED 'term' property is found in the content of each xhtml:dt element. The content of the xhtml:dt element MUST NOT contain child

elements. Terms are also known as 'tags'.

The OPTIONAL 'label' property is found in an xhtml:span element directly contained by each xhtml:dd element. The xhtml:span element MUST have 'label' as one of its 'class' values. There MUST NOT be more than one such xhtml:span within each xhtml:dd element.

The OPTIONAL 'scheme' property is found in an xhtml:a element

directly contained by each `xhtml:dd` element. The `xhtml:a` element MUST have 'scheme' as one of its 'rel' values. There MUST NOT be more than one such `xhtml:a` within each `xhtml:dd` element.

Each definition MAY contain an `xhtml:a` element with a 'rel' value of 'hCat'. This URI provided by the value of the element's 'href' attribute indicates a URI that responds to a simple protocol for editing of categories, which is described in the next section.

### [3.1](#) Creating and Editing hCat Categories

In addition to listing categories provided by a server, hCat XHTML documents allow conforming clients to edit and add categories. The server MAY communicate a URI where it accepts new categories by including an `xhtml:form` element with a 'class' attribute of 'hCat' in the hCat XHTML document. There MUST NOT be more than one `xhtml:form` element containing a class of 'hCat'. It would be preferable to use an 'id' attribute in this case, but many implementations depend on a generated value for a form's 'id' attribute. The `xhtml:form` element MUST be contained by the `xhtml:body` element, but MUST NOT be contained by an hCat definition list.

The values provided by the attributes of the `xhtml:form` element provide guidance for implementations wishing to submit content to the server. Servers MUST accept the MIME-type 'application/x-www-form-urlencoded', which is the default value for `xhtml:forms`.

- o As in XHTML, the 'action' attribute indicates the URI where new categories are submitted.
- o The 'accept-charset' attribute of the `xhtml:form` element MUST be present, and the list of accepted character sets SHOULD include 'utf-8'.
- o The value of the 'method' attribute MUST NOT be 'get', and SHOULD be 'post'. Naive user agents will likely treat other values as if they were 'get'.

The 'application/x-www-form-urlencoded' MIME-type is a name/value pair format. Servers identify their preferred name for each atom:

category property using `xhtml:input` elements contained by the hCat



xhtml:form element. The property is identified by the 'id' attribute of the xhtml:input element. For example,

```
<input name="foo" id="term" type="text" />
```

would indicate that the 'term' property is to be sent as the value of the 'foo' parameter.

The information provided by the hCat xhtml:form also applies to editing categories. To update a category, clients can send the parameters indicated by the form in an HTTP PUT request directed to the hCat URI conveyed by the relevant xhtml:a element in each xhtml:dd element. HTTP DELETE requests sent to that URI remove the category. This specification does not define a response to HTTP GET for such URIs, but servers SHOULD provide one (see [\[WEBARCH\]](#)). Servers MAY extend the hCat protocol with information in that response. This specification does not define a response to HTTP POST, but other specifications might do so. For example, a system supporting hierarchical categories might use HTTP POST requests to append a new category as a sub-category.

If the server encounters an error condition, the response body SHOULD be an hError XHTML document ([Section 4](#)).

#### [4.](#) hError: Error Documents

HTTP provides response codes which indicate the success or failure of a given request, but does not go into great detail on textual diagnostics for the end-user (see [\[RFC3117\]](#), [Section 3.3](#)). hError is an XHTML profile that encodes error information intended for the end-user.

hCat XHTML documents MUST indicate their profile in the 'head' element. The profile URI for hCat documents is "http://www.example.org/2005/Atom/hError" [example URI--do not deploy].

hCat provides three locations for error information:

1. The `xhtml:title` element MUST be present, and contains a short description of the error. Longer titles risk truncation due to GUI constraints.
2. An `xhtml:p` element with an 'id' attribute value of 'hError' MAY be present, and contains a longer description of the issue. The hError `xhtml:p` element MAY contain child elements, but many user interfaces will not display HTML.
3. An `xhtml:span` element with an 'id' attribute value of 'hErrorId' MAY be present, and contains a identifier for the error. This error identifier could be a unique identifier useful for locating a transaction during support requests, or a more general code identifying a specific condition.

An example hError XHTML document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://www.example.org/2005/Atom/hError">
    <title>We're Sorry! File not found.</title>
  </head>
  <body>
    <p id="hError">
      Hmm, can't find that entry.
      You can contact an administrator for help:
      admin@example.com.
    </p>
    <p>
      The unique identifier for this error is
      <span id="hErrorId">574DDDF2-A1E1-4898-B21B-EBB2DD16B38C</span>.
    </p>
  </body>
</html>
```

## [5.](#) Security Considerations

[[anchor5: What isn't a consideration with this? ...will fix]]

## [6.](#) References

### [6.1](#) Normative References

- [APP] Gregorio, J. and B. de hOra, "The Atom Publishing Protocol", work-in-progress, July 2005.
- [AtomFormat] Nottingham, M. and R. Sayre, "The Atom Syndication Format", IESG Approved, awaiting RFC number, July 2005.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [XHTML] Pemberton, S., "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)", W3C REC REC-xhtml1-20020801, August 2002, <<http://www.w3.org/TR/2002/REC-xhtml1-20020801>>.

### [6.2](#) Informative References

- [RFC3117] Rose, M., "On the Design of Application Protocols", [RFC 3117](#), November 2001.
- [WEBARCH] Walsh, N. and I. Jacobs, "Architecture of the World Wide

Web, Volume One", W3C REC REC-webarch-20041215,  
December 2004,  
<<http://www.w3.org/TR/2004/REC-webarch-20041215>>.

Sayre

Expires March 7, 2006

[Page 12]

---

Internet-Draft

XHTML Microformats for the APP

September 2005

URIs

[1] <<http://www.imc.org/atom-protocol/index.html>>

Author's Address

Robert Sayre

Email: [rfsayre@boswijck.com](mailto:rfsayre@boswijck.com)

URI: <http://boswijck.com>

#### [Appendix A](#). An Example hCat and hError Server

You can run this server from the command line using the Python programming language <<http://www.python.org>>.

Syntax: "python this\_script.py"

This will start a server running at 'http://localhost:8888'. You can visit in your browser, and try the client in [Appendix B](#).

On Microsoft Windows, the server may not respond to an interrupt

command (^Z CR) immediately. Visit the server one more time after attempting to interrupt, and the server will quit.

Patches welcome :)

```
import BaseHTTPServer, cgi
```

```
categories = [{ 'term':u'\u201CHello World!\u201D',
                 'label':u'Not much to describe',
                 'scheme':None},
               { 'term':u'Hello World!',
                 'label':None,
                 'scheme':None},
               { 'term':u'Hello World!',
                 'label':u'Not much to describe',
                 'scheme':None},
               { 'term':u'Hello World!',
                 'label':u'An example label',
                 'scheme':u'http://example.com'}]
```

```
hcat_template = u"""
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>
<html xmlns='http://www.w3.org/1999/xhtml'>
  <head profile='http://www.example.org/2005/Atom/hCat'>
    <title>hCat Demo</title>
  </head>
  <body>

    <h1>hCat Category Demo</h1>
    <p><em>Warning:</em> requires
      I\u00F1t\u00EBrn\u00E2ti\u00F4n\u00E0liz\u00E6ti\u00F8n</p>

    <form class='hCat' action='add'
      method='post' accept-charset='utf-8'>
      <fieldset>
        Term: <input name='term' id='term' type='text' /><br />
```

```
Label: <input name='label' id='label' type='text' /><br />
Scheme: <input name='scheme' id='scheme' type='text' /><br />
<input type='submit' value='Add' />
</fieldset>
```



```

        </form>

        <dl class='category'>
%s
        </dl>
    </body>
</html>
"""

herror_template = u"""
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>
<html xmlns='http://www.w3.org/1999/xhtml'>
    <head profile='http://www.example.org/2005/Atom/hError'>
        <title>Error %(code)d: %(shortDesc)s</title>
    </head>
    <body>
        <p>Error %(code)d: %(shortDesc)s</p>
        %(hError)s
        %(hErrorId)s
    </body>
</html>
"""

cat_template = u"""
Term: %(term)s
Label: %(label)s
Scheme: %(scheme)s"""

class HCatHandler(BaseHTTPServer.BaseHTTPRequestHandler):

    def do_GET(self):
        if self.path == "/":
            self.send_response(200)
            self.send_header('Content-type',self._gen_ctype_header())
            self.end_headers()
            resp = hcat_template % self.get_cats()
            self.wfile.write(resp.encode('utf-8'))
        elif self._is_cat(self.path):
            cat = categories[int(self.path[8])]
            self.send_response(200)
            self.send_header('Content-Type',
                            'text/plain; charset=utf-8')
            self.end_headers()

```

```
        resp = cat_template % cat
        self.wfile.write(resp.encode('utf-8'))
    else:
        self.send_herror(404, "File not found",
                        "Are you sure the address is right?",
                        self.gen_err_id())

def do_DELETE(self):
    if (self._is_cat(self.path)
        and categories[int(self.path[8])]):
        categories[int(self.path[8])] = None
        self.send_response(204)
        self.end_headers()
    else:
        self.send_herror(404, "File not found",
                        "Category does not exist",
                        self.gen_err_id())

def do_PUT(self):
    data = self._read_data()
    if not data:
        return
    if (self._is_cat(self.path)
        and categories[int(self.path[8])]):
        form = cgi.parse_qs(data)
        if not (form.has_key("term")):
            self.send_herror(500, "Error",
                            "Term is required.",
                            self.gen_err_id())
        return
        categories[int(self.path[8])] = self.get_cat(form)
        self.send_response(204)
        self.end_headers()
    else:
        self.send_herror(404, "File not found",
                        "Category does not exist",
                        self.gen_err_id())

def do_POST(self):
    redirect = "/"
    host = self.headers.getheader('host')
    if(host):
        redirect = "http://" + host + redirect
    data = self._read_data()
    if not data:
        return
    if self.path == "/add":
        form = cgi.parse_qs(data)
```

```
if not (form.has_key("term")):
```

Internet-Draft

XHTML Microformats for the APP

September 2005

```
        self.send_herror(500, "Error",
                          "Term is required.",
                          self.gen_err_id())
        return
    categories.append(self.get_cat(form))
    self.send_response(303)
    self.send_header('Location', redirect)
    self.end_headers()
else:
    self.send_herror(404, "POSTed to incorrect location",
                    "Are you sure the address is right?",
                    self.gen_err_id())

def _read_data(self):
    con_type = self.headers.getheader('content-type')
    if con_type != 'application/x-www-form-urlencoded':
        msg = "The content type was %s" % con_type
        self.send_herror(415, "Unsupported Content-Type", msg,
                        self.gen_err_id())
        return
    clen = self.headers.getheader('content-length')
    if clen:
        clen = int(clen)
    else:
        self.send_herror(411, "Missing Content-Length", None,
                        self.gen_err_id())
        return
    return self.rfile.read(clen)

def _is_cat(self, path):
    if (path.startswith('/ed?cat=') and int(path[8]) >= 0
        and int(path[8]) < len(categories)):
        return True
    else:
        return False

def _gen_ctype_header(self):
    # we want to send XHTML, but will fudge with
    # text/html if necessary (IE)
    # TODO: parse these correctly
```

```

prefer = 'application/xhtml+xml'
template = '%s; charset=utf-8'
accept = self.headers.get('Accept')
if accept:
    if prefer in accept:
        ctype = prefer
    else:
        ctype = 'text/html'

```

```

else:
    ctype = prefer
    return '%s; charset=utf-8' % ctype

def get_cat(self, form):
    def find_in_form(k):
        if form.has_key(k):
            return unicode(form[k][0], 'utf-8')
        else:
            return None

    return {'term': find_in_form('term'),
            'label': find_in_form('label'),
            'scheme': find_in_form('scheme')}

# loop through the categories in memory and return dt/dd pairs
def get_cats(self):
    res = u""
    for i in range(len(categories)):
        if categories[i]:
            cat = categories[i]
            res += u"    <dt>%s</dt>\n" % cat['term']
            res += u"    <dd>\n"
            if cat['label']:
                res += u"        <span class='label'>"
                res += cat['label']
                res += u"</span>\n"
            res += u"        <a href='/ed?cat=%d'" % i
            res += u" rel='hCat'>Edit</a>\n"
            if cat['scheme']:
                res += u"        <a href='"
                res += cat['scheme']
                res += u"' rel='scheme'>"

```

```

        res += u"</a>\n"
        res += u"    </dd>\n\n"

    return res

def send_herror(self, code, short_desc,
                long_desc=None, err_id=None):

    # log with the base class facilities
    try:
        short, long = self.responses[code]
    except KeyError:
        short, long = '???', '???'
    message = "%s id: %s" % (short, err_id)
    self.log_error("code %d, message %s", code, message)

```

Sayre

Expires March 7, 2006

[Page 18]

---

Internet-Draft

XHTML Microformats for the APP

September 2005

```

    # prepare hError doc
    if long_desc:
        hError = "<p id='hError'>%s</p>" % long_desc
    else:
        hError = ''
    if err_id:
        hErrorId = ""<p>The identifier for this error is:
        <span id='hErrorId'>%s</span></p>"" % err_id
    else:
        hErrorId = ''
    content = (herror_template %
               {'code': code,
                'shortDesc': short_desc,
                'hError': hError,
                'hErrorId': hErrorId})

    self.send_response(code, short_desc)
    self.send_header("Content-Type", "text/html")
    self.send_header('Connection', 'close')
    self.end_headers()
    if self.command != 'HEAD' and (code >= 200 and
                                   code not in (204, 304)):
        self.wfile.write(content)

def gen_err_id(self):
    from random import choice

```

```
lnd='0123456789'
return ''.join(map(lambda x,y=lnd: choice(y), range(10)))

PORT = 8888

httpd = BaseHTTPServer.HTTPServer(("", PORT), HCatHandler)
print "serving at port", PORT
httpd.serve_forever()
```

## [Appendix B](#). An Example hCat and hError Client

You can run this client from the command line using the Python programming language <<http://www.python.org>>. A sample server is included in [Appendix A](#).

Syntax: "python this\_script.py <http://example.com>"

This script uses a SAX handler <<http://www.saxproject.org>> in an effort to test viability in performance constrained environments.

Patches welcome :)

```
import sys,os,urllib2,urllib,urlparse,httplib
import xml.sax
from xml.sax.handler import *
import cStringIO
```

```
class Category:
```



```

        (XHTML_NS,u'body'):self.open_body},
IN_HEAD:  {(XHTML_NS,u'head',CLOSE):self.close_head},
IN_BODY:  {(XHTML_NS,u'body',CLOSE):self.close_body,
            (XHTML_NS,u'dl'):self.open_dl,
            (XHTML_NS,u'form'):self.open_form},
IN_DL:    {(XHTML_NS,u'dl',CLOSE):self.close_dl,
            (XHTML_NS,u'dt'):self.open_dt,
            (XHTML_NS,u'dd'):self.open_dd},
IN_DT:    {(XHTML_NS,u'dt',CLOSE):self.close_dt},
IN_DD:    {(XHTML_NS,u'dd',CLOSE):self.close_dd,
            (XHTML_NS,u'span'):self.open_span,
            (XHTML_NS,u'a'):self.open_anchor,
            (XHTML_NS,u'dl'):self.open_dl},
IN_LABEL: {(XHTML_NS,u'span',CLOSE):self.close_span},
IN_FORM:  {(XHTML_NS,u'form',CLOSE):self.close_form,
            (XHTML_NS,u'input'):self.open_input,
            (XHTML_NS,u'dl'):self.open_dl}
    }

def open_html(self, name, attrs):
    return IN_HTML

def close_html(self, name):
    return END

def open_head(self, name, attrs):
    profile = attrs.get((None,u'profile'))
    if profile and HCAT_PROFILE in profile.split():
        return IN_HEAD
    else:
        return END

def close_head(self, name):
    return IN_HTML

def open_body(self, name, attrs):

```

```

        return IN_BODY

def close_body(self, name):
    return IN_HTML

```



```

def open_dl(self, name, attrs):
    if attrs.get((None,u'class'))==u'category':
        self._lists.append(self.state)
        return IN_DL
    else:
        return None

def close_dl(self, name):
    return self._lists.pop(-1)

def open_dt(self, name, attrs):
    self._buf = ""
    return IN_DT

def close_dt(self, name):
    cat = Category(self._buf)
    self.cats.append(cat)
    return IN_DL

def open_dd(self, name, attrs):
    self._buf = ""
    return IN_DD

def close_dd(self, name):
    return IN_DL

def open_span(self, name, attrs):
    if attrs.get((None,u'class'))==u'label':
        self._buf=''
        return IN_LABEL
    else:
        return None

def close_span(self, name):
    self.cats[-1].label = self._buf
    return IN_DD

def open_form(self, name, attrs):
    class_att = attrs.get((None,u'class'))
    if class_att and u'hCat' in class_att.split():
        self.form = HCatForm()
        uri = attrs.get((None,u'action'))
        if uri:

```

```

        self.form.uri = urlparse.urljoin(self._base,uri)
        csets = attrs.get((None,u'accept-charset'))
        # charsets are 'space and/or comma separated' :/
        csets = [x for subseq in
                    [x.split(',') for x in csets.split()]
                    for x in subseq if x != '']
        self.form.charsets = csets
        return IN_FORM
    else:
        return None

def close_form(self, name):
    return IN_BODY

def open_input(self, name, attrs):
    params = ('term','scheme','label')
    input_id = attrs.get((None,u'id'))
    if input_id:
        name = attrs.get((None,u'name'))
        if input_id=='term':
            self.form.term = name
        elif input_id=='label':
            self.form.label = name
        elif input_id=='scheme':
            self.form.scheme = name
    return None

def open_anchor(self, name, attrs):
    rel = attrs.get((None,u'rel'))
    href = attrs.get((None,u'href'))
    if rel and u'scheme' in rel.split():
        self.cats[-1].scheme = href
    elif rel and u'hCat' in rel.split():
        self.cats[-1].uri = urlparse.urljoin(self._base,href)
    return None

## SAX Events
def startElementNS(self, name, qname, attrs):
    try:
        trans_func = self.transitions[self.state][name]
        self.state = trans_func(name,attrs) or self.state
    except KeyError:
        pass

def endElementNS(self, name, qname):
    try:
        event = (name[0],name[1],CLOSE)
        trans_func = self.transitions[self.state][event]

```

Internet-Draft

XHTML Microformats for the APP

September 2005

```
        self.state = trans_func(name) or self.state
    except KeyError:
        pass

    def characters(self, chars):
        self._buf += chars

class ErrorHandler(ContentHandler):
    def __init__(self):
        self.state = START
        self._buf = ''

        self.title = None
        self.hError = None
        self.hErrorId = None

        self.transitions = {
            START:      {(XHTML_NS,u'html'):self.open_html},
            IN_HTML:    {(XHTML_NS,u'html',CLOSE):self.close_html,
                        (XHTML_NS,u'head'):self.open_head,
                        (XHTML_NS,u'body'):self.open_body},
            IN_HEAD:   {(XHTML_NS,u'head',CLOSE):self.close_head,
                        (XHTML_NS,u'title'):self.open_title},
            IN_TITLE:  {(XHTML_NS,u'title',CLOSE):self.close_title},
            IN_BODY:   {(XHTML_NS,u'body',CLOSE):self.close_body,
                        (XHTML_NS,u'p'):self.open_p,
                        (XHTML_NS,u'span'):self.open_span},
            IN_P:      {(XHTML_NS,u'p',CLOSE):self.close_p},
            IN_SPAN:   {(XHTML_NS,u'span',CLOSE):self.close_span},
        }

    def open_html(self, name, attrs):
        return IN_HTML

    def close_html(self, name):
        return END

    def open_head(self, name, attrs):
        profile = attrs.get((None,u'profile'))
        if profile and HERROR_PROFILE in profile.split():
            return IN_HEAD
        else:
```

```
        return END

    def close_head(self, name):
        return IN_HTML

    def open_title(self, name, attrs):
```

```
        self._buf = ''
        return IN_TITLE

    def close_title(self, name):
        self.title = self._buf
        return IN_HEAD

    def open_body(self, name, attrs):
        return IN_BODY

    def close_body(self, name):
        return IN_HTML

    def open_p(self, name, attrs):
        if attrs.get((None,u'id'))==u'hError':
            self._buf = ''
            return IN_P
        else:
            return None

    def close_p(self, name):
        self.hError = self._buf
        return IN_BODY

    def open_span(self, name, attrs):
        if attrs.get((None,u'id'))==u'hErrorId':
            self._buf = ''
            return IN_SPAN
        else:
            return None

    def close_span(self, name):
        self.hErrorId = self._buf
        return IN_BODY
```

```

## SAX Events
def startElementNS(self, name, qname, attrs):
    try:
        trans_func = self.transitions[self.state][name]
        self.state = trans_func(name,attrs) or self.state
    except KeyError:
        pass

def endElementNS(self, name, qname):
    try:
        event = (name[0],name[1],CLOSE)
        trans_func = self.transitions[self.state][event]
        self.state = trans_func(name) or self.state

```

```

        except KeyError:
            pass

def characters(self, chars):
    self._buf += chars

def prompt(str):
    if str[-1:] != ' ': str=str+' '
    if len(str) > 40:
        print str
        sys.stdout.flush()
        return raw_input('--> ')
    else:
        return raw_input(str)

class HCatEditor:
    def __init__(self, uri):
        self._cats = []
        self.uri = uri
        self._form = None

    def refresh_cats(self):
        request = urllib2.Request(self.uri)
        try:
            data = self.make_request(request).read()
            self.parse(data)

```

```

except urllib2.HTTPError,e:
    self.print_error(e.read())

def parse(self,data):
    hCat_handler = HCatHandler(self.uri)
    parser = xml.sax.make_parser()
    parser.setFeature(xml.sax.handler.feature_namespaces, 1)
    parser.setContentHandler(hCat_handler)
    # you may want to provide a local resolver for
    # the DTDs, so you don't hit w3.org
    # parser.setEntityResolver(XhtmlResolver())
    parser.parse(cStringIO.StringIO(data))
    self._cats=hCat_handler.cats
    self._form=hCat_handler.form
    print '\n---- Server Categories ----'
    for i in range(len(self._cats)):
        print '[%d]  %s' % (i+1,self._cats[i])
    print '-----\n'

def make_request(self, request):

```

```

request.add_header('User-agent',
                   'hCatDemo/00 +http://franklinmint.fm')
accept_list = 'application/xhtml+xml,text/html,'
accept_list += 'text/plain;q=0.9,*/*;q=0.5'
request.add_header('Accept', accept_list)
opener = urllib2.build_opener()
return opener.open(request)

def enter_field(self,noun,current):
    if(current):
        curr = current.encode(UE)
        new = prompt('Enter a %s [%s]' % (noun,curr))
    else:
        new = prompt('Enter a %s' % noun)
    if new != '':
        return new
    else:
        return current

def enter_fields(self, term=None, label=None, scheme=None):
    params = {}

```

```

term = unicode(self.enter_field('term',term),UE)
if self._form.label:
    label = self.enter_field('label',label)
    if label is not None:
        label = unicode(label,UE)
if self._form.scheme:
    scheme = self.enter_field('scheme',scheme)
    if scheme is not None:
        scheme = unicode(scheme,UE)
if label:
    params[self._form.label] = label.encode('utf-8')
if scheme:
    params[self._form.scheme] = scheme.encode('utf-8')
params[self._form.term] = term.encode('utf-8')
return params

def choose_add(self):
    if not self._form:
        print "No add capability"
        return
    if not 'utf-8' in self._form.charsets:
        print "I can only use utf-8 for this demo..."
        print "the form provided:",self._form.charsets
        return
    params = self.enter_fields()
    request = urllib2.Request(self._form.uri,
                              urllib.urlencode(params))

```

```

resp = self.make_request(request)
self.uri = resp.url
self.parse(resp.read())

def pick_cat(self):
    try:
        cat_num = int(prompt('Pick a number from the list')) - 1
        if cat_num < 0 or cat_num >= len(self._cats):
            raise ValueError
    except ValueError:
        print 'Value not in list.'
        return None
    if not self._cats[cat_num].uri:
        print "Category has no editing capability"

```

```

        return None
    return cat_num

def choose_delete(self):
    cat_num = self.pick_cat()
    if cat_num is not None:
        parts = urlparse.urlsplit(self._cats[cat_num].uri)
        conn = httplib.HTTPConnection(parts[1])
        conn.request('DELETE', parts[2]+'?' + parts[3])
        response = conn.getresponse()
        if response.status in range(200,299):
            print "\nOK"
            self.refresh_cats()
        elif response.status in range(400,599):
            data = response.read()
            self.print_error(data)
        else:
            print response.reason

    return

def choose_edit(self):
    cat_num = self.pick_cat()
    if cat_num is not None:
        label = self._cats[cat_num].label
        scheme = self._cats[cat_num].scheme
        if label:
            label = label.encode(UE)
        if scheme:
            scheme = scheme.encode(UE)
        term = self._cats[cat_num].term
        p = self.enter_fields(term.encode(UE),
                               label, scheme)
        body = urllib.urlencode(p)

```

```

headers = {'Content-Type':FORM_ENCODED,
           'Content-Length':len(body)}
parts = urlparse.urlsplit(self._cats[cat_num].uri)
conn = httplib.HTTPConnection(parts[1])
conn.request('PUT', parts[2]+'?' + parts[3],
             body, headers)
response = conn.getresponse()

```



```

        if response.status in range(200,299):
            print "\nOK"
            self.refresh_cats()
        elif response.status in range(400,599):
            data = response.read()
            self.print_error(data)
        else:
            print response.reason
    return

def print_error(self,data):
    hError_handler = HErrorHandler()
    parser = xml.sax.make_parser()
    parser.setFeature(xml.sax.handler.feature_namespaces, 1)
    parser.setContentHandler(hError_handler)
    # you may want to provide a local resolver for
    # the DTDs, so you don't hit w3.org.
    # parser.setEntityResolver(XhtmlResolver())
    print "An Error occured.\n"
    try:
        parser.parse(cStringIO.StringIO(data))
        if hError_handler.title:
            print hError_handler.title
        if hError_handler.hError:
            print "Description:"
            print hError_handler.hError
        if hError_handler.hErrorId:
            print "ID:" + hError_handler.hErrorId
    except xml.sax.SAXException, e:
        pass
    print ""

def usage():
    print "Usage:  %s http_uri" % os.path.basename(sys.argv[0])

def main():
    args = sys.argv[1:]
    if "-h" in args or "--help" in args or len(args)==0:
        usage()
        sys.exit(2)
    editor = HCatEditor(args[0])

```

```
editor.refresh_cats()
msg = "Choose: [a]dd, [e]dit, [d]elete, [r]efresh, [q]uit"
while True:
    x = prompt(msg)
    if x == 'q':
        return 0
    elif x == 'r':
        editor.refresh_cats()
    elif x == 'e':
        editor.choose_edit()
    elif x == 'd':
        editor.choose_delete()
    elif x == 'a':
        editor.choose_add()
    else:
        print 'Unknown option.\n'

if __name__ == "__main__":
    main()
```

Internet-Draft

XHTML Microformats for the APP

September 2005

## Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

## Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Sayre

Expires March 7, 2006

[Page 31]

---

Internet-Draft

XHTML Microformats for the APP

September 2005

#### Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

Sayre

Expires March 7, 2006

[Page 32]