Network Working Group
Request for Comments: 2936
Category: Informational

D. Eastlake
Motorola
C. Smith
Royal Bank of Canada
D. Soroka
IBM

September 2000

HTTP MIME Type Handler Detection

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

Entities composing web pages to provide services over the Hypertext Transfer Protocol (HTTP) frequently have the problem of not knowing what Multipurpose Internet Mail Extensions (MIME) types have handlers installed at a user's browser. For example, whether an Internet Open Trading Protocol (IOTP) or VRML or SET or some streaming media handler is available. In some cases they would want to display different web pages or content depending on a MIME handler's availability. This document summarizes reasonable techniques to solve this problem for most of the browsers actually deployed on the Internet as of early 2000. It is intended to be of practical use to implementors during the period before the wide deployment of superior standards based techniques which may be developed.

Acknowledegements

Helpful comments by Tony Lewis of Visa have been incorporated.

Table of Contents

1. Introduction

Entities composing web pages to provide services over [HTTP] frequently have the problem of not knowing what [MIME] types have handlers installed at a user's browser. For example, whether an [IOTP] or VRML or [SET] or some streaming media handler is available. In many cases they would want to display different web pages or content depending on a MIME handler's availability. Sending a response with a MIME type that is not supported frequently results in interrupting the flow of the user experience, browser queries as to what to do with the data being provided, and, of course, failure to provide the behavior that would have occurred had the correct MIME type handler been installed.

This document describes reasonable techniques to solve this problem for most of the browsers actually deployed on the Internet as of early 2000. It is intended to be of practical use to implementors during the period before the wide deployment of superior standards based techniques which may be developed. It is written in terms of determining whether a handler for application/iotp or application/x-iotp exists but is equally applicable to other MIME types.

2. The HTTP 'Accept' Header

The problem should be solved by the Hyper Text Transport Protocol [HTTP] request "Accept" header which lists accepted [MIME] types. This header is present in both Version 1.0 and 1.1 of HTTP and its content is supposed to be a list of MIME types and subtypes that are accepted. The only problem is that many browsers just send "*/*" or the like.

If the particular MIME type you are looking for is specifically present in the Accept header, it is generally safe to assume that a handler for it is actually installed or part of the browser.

NOTE: Although not part of the main topic of this document, if you are designing MIME type handler software and have access to a browser interface that allows you to request the insertion of the MIME type or types your software handles into the Accept header, you generally should do so. It will make it easier for servers sensitive to that MIME type to respond correctly.

3. JavaScript

Most recent browsers support one or more scripting languages of which the most widely deployed is "JavaScript". These scripting languages appear in web pages and permit the interpretive execution of programming language constructs that can probe the browser environment, conditionally cause different page contents to be displayed, etc. For example, Appendix A shows JavaScript available from the Netscape web site for determining what operating system, browser, and version on which a web page is appearing.

NOTE: JavaScript is a trademark of SUN Microsystems, Inc. It was originally called LiveScript. It has nothing to do with the Java language.

The syntax for script use appears to be a Hyper Text Markup Language (HTML) comment so that browsers that do not support scripting will ignore such items. That is, script use is preceded by "<!--" and terminated by "-->". The following is a simple example of conditional execution of parts of a web page based on JavaScript MIME type handler detection.

```
<SCRIPT LANGUAGE=JAVASCRIPT>
<!-- hide it
if (navigator.mimeTypes && navigator.mimeTypes.length > 0) {
  if ( navigator.mimeTypes["application/iotp"] ||
       navigator.mimeTypes["application/x-iotp"]) {
    // here if IOTP handler exists
    }
  else {
    // here if IOTP handler does not exist
    }
  }
// end and hide -->
</SCRIPT>
```

4. ActiveX and the Windows Registry

If running on Microsoft Windows Internet Explorer version 3 or 4, it is necessary to query the Windows Registry to determine local MIME type support. Although these broswers support JavaScript, in v3 the mimeTypes array is not present and in v4 the mimeTypes array is present but always empty. For example, executing the following code will test for support of the IOTP types:

```
CString iotpString, xiotpString;
char* Key, Keyx;

   int rc, rcx;
   iotpString =
"SOFTWARE\Classes\MIME\Database\Content Type\application/iotp";
   xiotpString =
"SOFTWARE\Classes\MIME\Database\Content Type\application/x-iotp";
   Key = iotpString.GetBuffer(1);
   Keyx = xiotpString.GetBuffer(1);
   rc = RegOpenKeyEx(HKEY_LOCAL_MACHINE, Key, 0, KEY_READ, hDefKey);
   rcx = RegOpenKeyEx(HKEY_LOCAL_MACHINE, Keyx, 0, KEY_READ, hDefKey);
if ( ( rc  == ERROR_SUCCESS ) || ( rcx == ERROR_SUCCESS ) )
 {
 // IOTP Handler exists
 }
else
 {
 // No IOTP Handler
 }
```

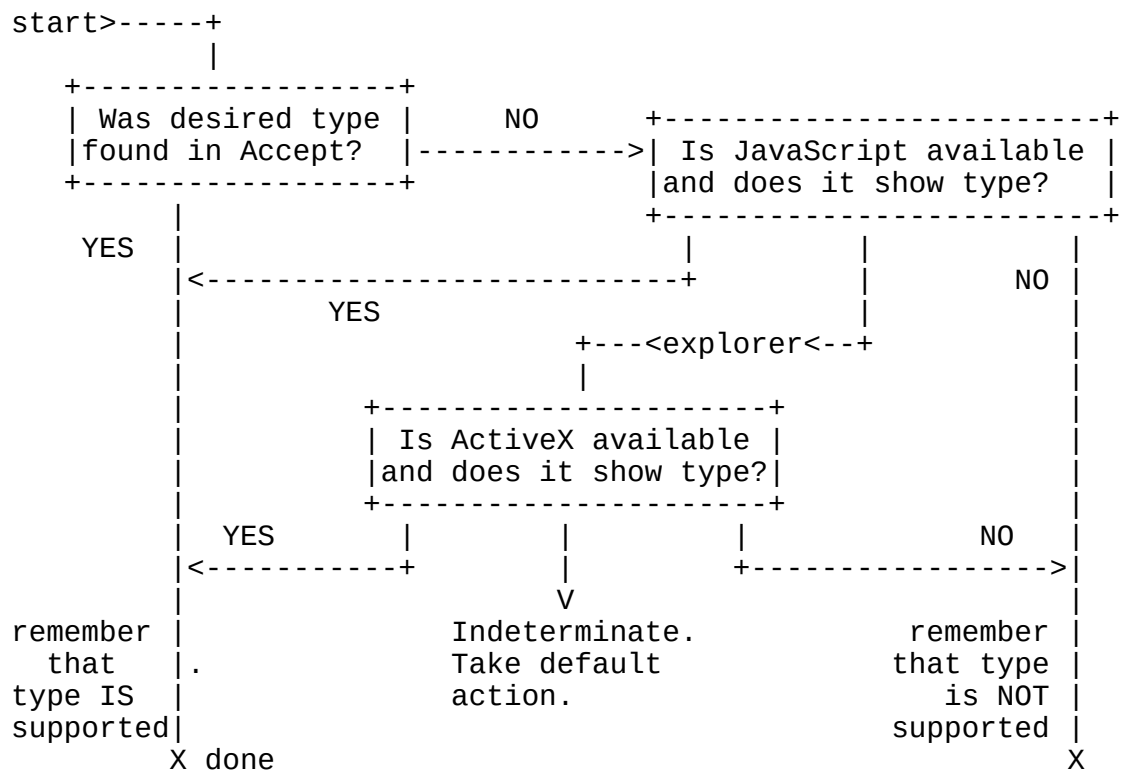
NOTE: ActiveX is a trademark of Microsoft and was originally called Sweeper.

5. ECML, The Electronic Commerce Modeling Language

A industry group has recently proposed a standard for fields used in electronic commerce. This fields allow "wallet" software acting for the consumer to convey standardized information to a merchant, including information as to what payment related protocols are supported at the customer site. See [ECML].

6. Putting It All Together

The following diagram indicates how these techniques can be put together.



7. Future Development

Active work is proceeding in the IETF, World Wide Web Consortium [W3C], and other standards and industry groups concerning content and capabilities negotiation. This work is likely to lead to superior methods to implement the functionality described herein. However, near universal deployment of such new standards/features will take some time. Thus you should expect the methods given herein to be obsoleted, but perhaps not for some time.

8. Security Considerations

It should be noted that the variety of ActiveX control suggested above is reading the user's registry, that is, examining their computer and reporting back some information it has discovered. This may be a concern among some users.

In general, the use of JavaScript and, even more so, ActiveX is dangerous because they are so powerful. JavaScript or ActiveX from a web page could be invisibly damaging to the client.

Security of web interactions is normally provided by adopting channel encryption on the browser to server connections, such as [TLS]. In the absence of some such additional security outside of HTTP, requests and/or responses may be forged or tampered with.

9. IANA Considerations

None specific to the techniques described herein. For MIME types and type registration procedures, see [MIME: RFCs 2046, 2048].

References

[ECML] Eastlake, D. and T. Goldstein, "ECML v1: Field Names for E-Commerce", RFC 2706, October 1999.

[HTTP] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[HTTP] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[IOTP] Burdett, D., "Internet Open Trading Protocol - IOTP Version 1.0", RFC 2801, April 2000.

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[MIME] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[MIME] Freed, N., Klensin, J. and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, November 1996.

[SET] "Secure Electronic Transaction (SET) Specification, Version 1.0", May 31, 1997, available from <http://www.setco.org>.
Book 1: Business Description
Book 2: Programmer's Guide
Book 3: Formal Protocol Definition

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[W3C] World Wide Web Consortium, <www.w3.org>

Appendix A: Browser Version Sniffer Code

```
<SCRIPT LANGUAGE="JavaScript">
<!-- hide JavaScript from non-JavaScript browsers
// Ultimate client-side JavaScript client sniff.
// (C) Netscape Communications 1999.
//     Permission granted to reuse and distribute.
// Revised 17 May 99 to add is_nav5up and is_ie5up (see below).

// Everything you always wanted to know about your JavaScript client
// but were afraid to ask. Creates "is_" variables indicating:
// (1) browser vendor:
//     is_nav, is_ie, is_opera
// (2) browser version number:
//     is_major (integer indicating major version number: 2, 3, 4 ...)
//     is_minor (float   indicating full  version number:
//                                              2.02, 3.01, 4.04 ...)
// (3) browser vendor AND major version number
//     is_nav2, is_nav3, is_nav4, is_nav4up, is_nav5, is_nav5up,
//     is_ie3, is_ie4, is_ie4up
// (4) JavaScript version number:
//     is_js (float indicating full JavaScript version number:
//                                                  1, 1.1, 1.2 ...)
// (5) OS platform and version:
//     is_win, is_win16, is_win32, is_win31,
//     is_win95, is_winnt, is_win98
//     is_os2
//     is_mac, is_mac68k, is_macppc
//     is_unix
//        is_sun, is_sun4, is_sun5, is_suni86
//        is_irix, is_irix5, is_irix6
//        is_hpux, is_hpux9, is_hpux10
//        is_aix, is_aix1, is_aix2, is_aix3, is_aix4
//        is_linux, is_sco, is_unixware, is_mpras, is_reliant
//        is_dec, is_sinix, is_freebsd, is_bsd
//     is_vms
//
// See http://www.it97.de/JavaScript/JS_tutorial/bstat/navobj.html and
// http://www.it97.de/JavaScript/JS_tutorial/bstat/Browseraol.html
// for detailed lists of userAgent strings.
//
// Note: you don't want your Nav4 or IE4 code to "turn off" or
// stop working when Nav5 and IE5 (or later) are released, so
// in conditional code forks, use is_nav4up ("Nav4 or greater")
// and is_ie4up ("IE4 or greater") instead of is_nav4 or is_ie4
// to check version in code which you want to work on future
// versions.
```

```
  // convert all characters to lowercase to simplify testing
  var agt=navigator.userAgent.toLowerCase();

  // *** BROWSER VERSION ***
  // Note: On IE5, these return 4, so use is_ie5up to detect IE5.
  var is_major = parseInt(navigator.appVersion);
  var is_minor = parseFloat(navigator.appVersion);

  // Note: Opera and WebTV spoof Navigator.  We do strict client
  //  detection.  If you want to allow spoofing, take out the tests
  //  for opera and webtv.
  var is_nav  = ((agt.indexOf('mozilla')!=-1)
              && (agt.indexOf('spoofer')==-1)
              && (agt.indexOf('compatible') == -1)
              && (agt.indexOf('opera')==-1)
              && (agt.indexOf('webtv')==-1));
  var is_nav2 = (is_nav && (is_major == 2));
  var is_nav3 = (is_nav && (is_major == 3));
  var is_nav4 = (is_nav && (is_major == 4));
  var is_nav4up = (is_nav && (is_major >= 4));
  var is_navonly      = (is_nav && ((agt.indexOf(";nav") != -1) ||
                        (agt.indexOf("; nav") != -1)) );
  var is_nav5 = (is_nav && (is_major == 5));
  var is_nav5up = (is_nav && (is_major >= 5));

  var is_ie   = (agt.indexOf("msie") != -1);
  var is_ie3  = (is_ie && (is_major < 4));
  var is_ie4  = (is_ie && (is_major == 4)
                       && (agt.indexOf("msie 5.0")==-1) );
  var is_ie4up  = (is_ie  && (is_major >= 4));
  var is_ie5  = (is_ie && (is_major == 4)
                       && (agt.indexOf("msie 5.0")!=-1) );
  var is_ie5up  = (is_ie  && !is_ie3 && !is_ie4);

  // KNOWN BUG: On AOL4, returns false if IE3 is embedded browser
  // or if this is the first browser window opened.  Thus the
  // variables is_aol, is_aol3, and is_aol4 aren't 100% reliable.
  var is_aol   = (agt.indexOf("aol") != -1);
  var is_aol3  = (is_aol && is_ie3);
  var is_aol4  = (is_aol && is_ie4);

  var is_opera = (agt.indexOf("opera") != -1);
  var is_webtv = (agt.indexOf("webtv") != -1);

  // *** JAVASCRIPT VERSION CHECK ***
  var is_js;
  if (is_nav2 || is_ie3) is_js = 1.0
  else if (is_nav3 || is_opera) is_js = 1.1
```

```
  else if ((is_nav4 && (is_minor <= 4.05)) || is_ie4) is_js = 1.2
  else if ((is_nav4 && (is_minor > 4.05)) || is_ie5) is_js = 1.3
  else if (is_nav5) is_js = 1.4
  // NOTE: In the future, update this code when newer versions of JS
  // are released. For now, we try to provide some upward compatibility
  // so that future versions of Nav and IE will show they are at
  // *least* JS 1.x capable. Always check for JS version compatibility
  // with > or >=.
  else if (is_nav && (is_major > 5)) is_js = 1.4
  else if (is_ie && (is_major > 5)) is_js = 1.3
  // HACK: no idea for other browsers;
  //       always check for JS version with > or >=
  else is_js = 0.0;

  // *** PLATFORM ***
  var is_win   = ( (agt.indexOf("win")!=-1) ||
                   (agt.indexOf("16bit")!=-1) );
  // NOTE: On Opera 3.0, the userAgent string includes "Windows 95/NT4"
  // on all Win32, so you can't distinguish between Win95 and WinNT.
  var is_win95 = ((agt.indexOf("win95")!=-1) ||
                  (agt.indexOf("windows 95")!=-1));

  // is this a 16 bit compiled version?
  var is_win16 = ((agt.indexOf("win16")!=-1) ||
             (agt.indexOf("16bit")!=-1) ||
             (agt.indexOf("windows 3.1")!=-1) ||
             (agt.indexOf("windows 16-bit")!=-1) );

  var is_win31 = ((agt.indexOf("windows 3.1")!=-1) ||
                  (agt.indexOf("win16")!=-1) ||
                  (agt.indexOf("windows 16-bit")!=-1));

  // NOTE: Reliable detection of Win98 may not be possible.
  // It appears that:
  //  - On Nav 4.x and before you'll get plain "Windows" in userAgent.
  //  - On Mercury client, the 32-bit version will return "Win98", but
  //    the 16-bit version running on Win98 will still return "Win95".
  var is_win98 = ((agt.indexOf("win98")!=-1) ||
                  (agt.indexOf("windows 98")!=-1));
  var is_winnt = ((agt.indexOf("winnt")!=-1) ||
                  (agt.indexOf("windows nt")!=-1));
  var is_win32 = (is_win95 || is_winnt || is_win98 ||
                  ((is_major >= 4) &&
                   (navigator.platform == "Win32")) ||
                  (agt.indexOf("win32")!=-1) ||
                  (agt.indexOf("32bit")!=-1));

  var is_os2   = ((agt.indexOf("os/2")!=-1) ||
```

```
                (navigator.appVersion.indexOf("OS/2")!=-1) ||
                (agt.indexOf("ibm-webexplorer")!=-1));

  var is_mac    = (agt.indexOf("mac")!=-1);
  var is_mac68k = (is_mac && ((agt.indexOf("68k")!=-1) ||
                              (agt.indexOf("68000")!=-1)));
  var is_macppc = (is_mac && ((agt.indexOf("ppc")!=-1) ||
                               (agt.indexOf("powerpc")!=-1)));

  var is_sun   = (agt.indexOf("sunos")!=-1);
  var is_sun4  = (agt.indexOf("sunos 4")!=-1);
  var is_sun5  = (agt.indexOf("sunos 5")!=-1);
  var is_suni86= (is_sun && (agt.indexOf("i86")!=-1));
  var is_irix  = (agt.indexOf("irix") !=-1);    // SGI
  var is_irix5 = (agt.indexOf("irix 5") !=-1);
  var is_irix6 = ((agt.indexOf("irix 6") !=-1) ||
                  (agt.indexOf("irix6") !=-1));
  var is_hpux  = (agt.indexOf("hp-ux")!=-1);
  var is_hpux9 = (is_hpux && (agt.indexOf("09.")!=-1));
  var is_hpux10= (is_hpux && (agt.indexOf("10.")!=-1));
  var is_aix   = (agt.indexOf("aix") !=-1);      // IBM
  var is_aix1  = (agt.indexOf("aix 1") !=-1);
  var is_aix2  = (agt.indexOf("aix 2") !=-1);
  var is_aix3  = (agt.indexOf("aix 3") !=-1);
  var is_aix4  = (agt.indexOf("aix 4") !=-1);
  var is_linux = (agt.indexOf("inux")!=-1);
  var is_sco   = (agt.indexOf("sco")!=-1) ||
                 (agt.indexOf("unix_sv")!=-1);
  var is_unixware = (agt.indexOf("unix_system_v")!=-1);
  var is_mpras    = (agt.indexOf("ncr")!=-1);
  var is_reliant  = (agt.indexOf("reliantunix")!=-1);
  var is_dec   = ((agt.indexOf("dec")!=-1) ||
         (agt.indexOf("osf1")!=-1) ||
         (agt.indexOf("dec_alpha")!=-1) ||
         (agt.indexOf("alphaserver")!=-1) ||
         (agt.indexOf("ultrix")!=-1) ||
         (agt.indexOf("alphastation")!=-1));
  var is_sinix = (agt.indexOf("sinix")!=-1);
  var is_freebsd = (agt.indexOf("freebsd")!=-1);
  var is_bsd = (agt.indexOf("bsd")!=-1);
  var is_unix  = ((agt.indexOf("x11")!=-1) || is_sun ||
                is_irix || is_hpux ||
                is_sco ||is_unixware || is_mpras || is_reliant ||
                is_dec || is_sinix || is_aix || is_linux ||
                is_bsd || is_freebsd);

  var is_vms   = ((agt.indexOf("vax")!=-1) ||
                  (agt.indexOf("openvms")!=-1));
```

```
  </SCRIPT>
```

Authors' Addresses

Donald E. Eastlake 3rd
Motorola
140 Forest Avenue
Hudson, MA 01749 USA

Phone: +1 978-562-2827(h)
       +1 508-261-5434(w)
Fax: +1 508-261-4447(w)
EMail: Donald.Eastlake@motorola.com

Chris J. Smith
Royal Bank of Canada
277 Front Street West
Toronto, Ontario M5V 3A4 CANADA

Phone: +1 416-348-6090
Fax: +1 416-348-2210
EMail: chris.smith@royalbank.com

David M. Soroka
IBM
Raleigh, NC

Phone: +1 919-486-2684
Fax: +1 919-543-4653
EMail: dsoroka@us.ibm.com

Full Copyright Statement

Acknowledgement

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