HTTP MIME Type Handler Detection

<draft-ietf-trade-mime-detector-02.txt>

Donald E. Eastlake 3rd
Chris J. Smith
David M. Soroka

Status of This Document

This draft is intended to become an Informational RFC. Distribution of this document is unlimited. Comments should be sent to the TRADE WG mailing list <ietf-trade@eListX.com> or to the authors.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. 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/1id-abstracts.txt

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

Abstract

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. 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.

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 present in the Accept header, it is generally safe to assume that some specific 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 programing 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 you are running on.

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 bowsers that do not support scripting will ignore such items. That is, script use if preceeded 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.appName=="netscape") {  // wise precaution
  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. Microsoft ActiveX and the Windows Registry

If running on Internet Explorer version 3 or 4, it is necessary to query to 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, Key, 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.

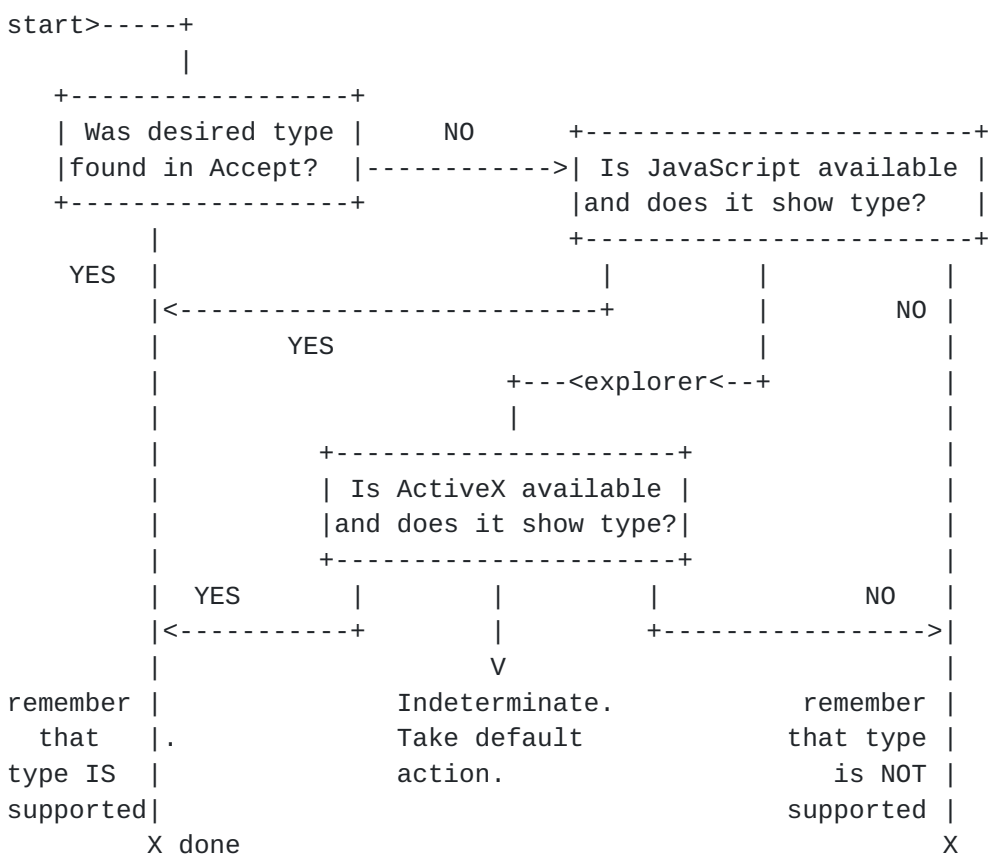
[TBD Stuff about ActiveX glue...?]

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 Electronic Commerce Modeling Language Alliance [ECML] 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 years.

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 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] <http://www.ecml.org>
RFC 2706 - "ECML v1: Field Names for E-Commerce", D. Eastlake, T. Goldstein, October 1999
draft-eastlake-ecom-fields2-*.txt, D. Eastlake, T. Goldstein.

[HTTP] RFC 1945 - "Hypertext Transfer Protocol -- HTTP/1.0", T. Berners-Lee, R. Fielding & H. Frystyk. May 1996.
RFC 2616 - "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999.

[IOTP] draft-ietf-trade-iotp-v1.0-protocol-*.txt - David Burdett, 2000.

[MIME] RFC 2045 - "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed & N. Borenstein. November 1996.
RFC 2046 - "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed & N. Borenstein. November 1996.
RFC 2047 - "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", K. Moore. November 1996.
RFC 2048 - "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", N. Freed, J. Klensin & J. Postel. 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] RFC 2246 - "The TLS Protocol Version 1.0", T. Dierks, C. Allen.

[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 1998.  Permission granted to reuse and
distribute.
   //Revised 20 April 98 to add is.nav4up and is.ie4up (see below).

   // Everything you always wanted to know about your JavaScript client
   // but were afraid to ask ... "Is" is the constructor function for "is"
object,
   // which has properties 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.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/obj_hierarchy/navobjFr.html
   //for a detailed list 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.

   function Is ()
```

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

// *** BROWSER VERSION ***
this.major = parseInt(navigator.appVersion)
this.minor = parseFloat(navigator.appVersion)
```

```
   this.nav  = ((agt.indexOf('mozilla')!=-1) &&
       ((agt.indexOf('spoofer')==-1) &&
       (agt.indexOf('compatible') == -1)))
   this.nav2 = (this.nav && (this.major == 2))
   this.nav3 = (this.nav && (this.major == 3))
   this.nav4 = (this.nav && (this.major == 4))
   this.nav4up = this.nav && (this.major >= 4)
   this.navonly  = (this.nav && (agt.indexOf(";nav") != -1))

   this.ie   = (agt.indexOf("msie") != -1)
   this.ie3  = (this.ie && (this.major == 2))
   this.ie4  = (this.ie && (this.major == 4))
   this.ie4up  = this.ie  && (this.major >= 4)

   this.opera = (agt.indexOf("opera") != -1)

   // *** JAVASCRIPT VERSION CHECK ***
   // Useful to workaround Nav3 bug in which Nav3
   // loads <SCRIPT LANGUAGE="JavaScript1.2">.
   if (this.nav2 || this.ie3) this.js = 1.0
   else if (this.nav3 || this.opera) this.js = 1.1
   else if (this.nav4 || this.ie4) this.js = 1.2
   // 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 1.2 capable. Always check for JS version compatibility
   // with > or >=.
   else if ((this.nav && (this.minor > 4.05)) || (this.ie && (this.major > 4)))
        this.js = 1.2
   else this.js = 0.0 // HACK: always check for JS version with > or >=

   // *** PLATFORM ***
   this.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.
   this.win95 = ((agt.indexOf("win95")!=-1) || (agt.indexOf("windows 95")!=-1))

   // is this a 16 bit compiled version?
   this.win16 = ((agt.indexOf("win16")!=-1)
       || (agt.indexOf("16bit")!=-1) || (agt.indexOf("windows 3.1")!=-1)
       || (agt.indexOf("windows 16-bit")!=-1) )
   this.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".
   this.win98 =
       ((agt.indexOf("win98")!=-1)||(agt.indexOf("windows 98")!=-1))
   this.winnt =
       ((agt.indexOf("winnt")!=-1)||(agt.indexOf("windows nt")!=-1))
   this.win32 = this.win95 || this.winnt || this.win98 ||
       ((this.major >= 4) && (navigator.platform == "Win32")) ||
       (agt.indexOf("win32")!=-1) || (agt.indexOf("32bit")!=-1)

   this.os2   = (agt.indexOf("os/2")!=-1)
       || (navigator.appVersion.indexOf("OS/2")!=-1)
       || (agt.indexOf("ibm-webexplorer")!=-1)

   this.mac= (agt.indexOf("mac")!=-1)
   this.mac68k = this.mac && ((agt.indexOf("68k")!=-1) ||
                 (agt.indexOf("68000")!=-1))
       this.macppc = this.mac && ((agt.indexOf("ppc")!=-1) ||
                 (agt.indexOf("powerpc")!=-1))
   this.sun   = (agt.indexOf("sunos")!=-1)
   this.sun4  = (agt.indexOf("sunos 4")!=-1)
   this.sun5  = (agt.indexOf("sunos 5")!=-1)
   this.suni86= this.sun && (agt.indexOf("i86")!=-1)
   this.irix  = (agt.indexOf("irix") !=-1)    // SGI
   this.irix5 = (agt.indexOf("irix 5") !=-1)
   this.irix6 =
       ((agt.indexOf("irix 6") !=-1) || (agt.indexOf("irix6")!=-1))
   this.hpux  = (agt.indexOf("hp-ux")!=-1)
   this.hpux9 = this.hpux && (agt.indexOf("09.")!=-1)
   this.hpux10= this.hpux && (agt.indexOf("10.")!=-1)
   this.aix   = (agt.indexOf("aix")  !=-1)  // IBM
   this.aix1  = (agt.indexOf("aix 1")  !=-1)
   this.aix2  = (agt.indexOf("aix 2")  !=-1)
   this.aix3  = (agt.indexOf("aix 3")  !=-1)
   this.aix4  = (agt.indexOf("aix 4")  !=-1)
   this.linux = (agt.indexOf("inux")!=-1)
   this.sco   = (agt.indexOf("sco")!=-1) || (agt.indexOf("unix_sv")!=-1)
   this.unixware = (agt.indexOf("unix_system_v")!=-1)
   this.mpras= (agt.indexOf("ncr")!=-1)
   this.reliant  = (agt.indexOf("reliantunix")!=-1)
   this.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)
   this.sinix = (agt.indexOf("sinix")!=-1)
   this.freebsd = (agt.indexOf("freebsd")!=-1)
```

```
   this.bsd = (agt.indexOf("bsd")!=-1)
```

```
   this.unix =
       (agt.indexOf("x11")!=-1) || this.sun || this.irix || this.hpux ||
       this.sco ||this.unixware || this.mpras || this.reliant ||
       this.dec || this.sinix || this.aix || this.linux || this.freebsd

   this.vms   = (agt.indexOf("vax")!=-1) || (agt.indexOf("openvms")!=-1)
   }

   var is;
   var isIE3Mac = false;
   // this section is designed specifically for IE3 for the Mac
   if ((navigator.appVersion.indexOf("Mac")!=-1) &&
           (navigator.userAgent.indexOf("MSIE")!=-1) &&
           (parseInt(navigator.appVersion)==3))
       isIE3Mac = true;
   else
       is = new Is();
   //--> end hide JavaScript
   </SCRIPT>
```

Authors Addresses

Donald E. Eastlake 3rd
Motorola
65 Shindegan Hill Road
Carmel, NY 10512 USA

Telephone: +1 914-276-2668(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

Telephone: +1 416-348-6090
FAX: +1 416-348-2210
email: chris.smith@royalbank.com

David M. Soroka
IBM
Raleigh, NC

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

Expiration and File Name

This draft expires September 2000.

Its file name is draft-ietf-trade-mime-detector-02.txt.