(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 May 2008 (08.05.2008)

PCT

(10) International Publication Number
WO 2008/055176 A1

(51) International Patent Classification:
G06F 17/22 (2006.01)

(21) International Application Number:
PCT/US2007/083048

(22) International Filing Date: 30 October 2007 (30.10.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/554,397 30 October 2006 (30.10.2006) US

(71) Applicant *(for all designated States except US)*: **GOOGLE INC.** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **TSUN, Stephen** [US/US]; 10475 Imperial Avenue, Cupertino, California 95014 (US).

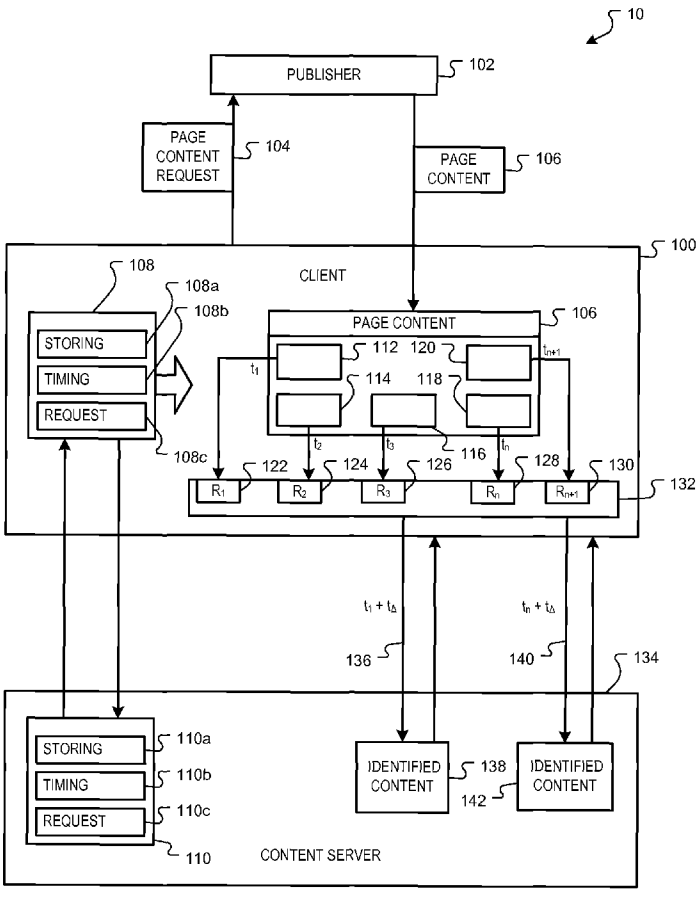
(74) Agent: **AARONSON, Lawrence A.**; Fish & Richardson P.c., P.O. Box 1022, Minneapolis, Minnesota 55440-1022 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: DIAGNOSTICS AND ERROR REPORTING FOR COMMON TAGGING ISSUES



(57) Abstract: Content requests are debugged in accordance with a presence of a flag in a request to a publisher. A document received from the publisher contains a script to debug requests for content to a content provider. The requests are examined to determine the presence of informational, warning and error conditions. The conditions are written to a debugging user interface.

WO 2008/055176 A1

Published:
— *with international search report*

DIAGNOSTICS AND ERROR REPORTING FOR COMMON TAGGING ISSUES

FIELD

This disclosure relates to content requests over a network.

BACKGROUND

Content displayed on web pages can be generated by one or more content servers in response to content requests. Publishers can embed content-server specific tags in web pages in order to serve content (e.g., ads) to their visitors from the content servers. Tags are used to describe what content slots exist in a given web page and what creatives can be served to each slot. Content may not appear in a given web page at the time for rendering due to network latencies, typographical errors in the tag names, and mis-configurations. For example, the tags may be incorrectly or incompletely specified, or correctly specified tags may not match the server-side settings. In addition, content presentation may fail due to client browser or operating system problems, locale or language issues, transient networking issues and the like.

When errors occur, a publisher typically will make a service call, which are time consuming and expensive to the content server provider. In many instances, the content server provider may have difficulty reproducing the underlying problem that causes an error. For example, the cause of the problem may be an older browser or non-supported version of an operating system. Transient network problems are also difficult to reproduce, which makes it difficult for the content server provider to diagnose the problem.

SUMMARY

Disclosed herein are systems and methods relating to the debugging of content requests. According to some implementations, a document at a location is requested that contains a script, where, the request includes an indicator. The requested document is received and the script is executed to test the status of a condition. The status is displayed in a window. The status can be indicated by color-coding in the window.

According to some implementations a system includes a content server configurable to receive content, a publisher server operatively coupled to the content server and configurable to determine a context from the content, and a content repository operatively coupled to the content server and configurable for providing the content server with content

associated with a context. The publisher server provides computer executable instructions to a client device upon receiving a request from the client device. The computer executable instructions are executed by the client device to determine conditions related to the communication of content from the content repository to the client device.

According to some implementations a system includes a processor configurable to request content from a remote location and an interface operatively coupled to the processor and configurable to display debugging information. The content includes computer executable instructions to determine the debugging information associated with the request, and the debugging information can includes errors associated with the display of the content in the interface.

These and other example implementations can include one or more of the following features or advantages. In some implementations, the debugging information is provided in a user interface that allows the publisher to diagnose a problem and provide an effective report to the content server provider.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a system for requesting content from a content server.

Fig. 2 is an example process for debugging a request for content from the content server.

Fig. 3 is another example process for debugging a request for content from the content server.

Figs. 4-6 are example interfaces for displaying debugging errors.

DETAILED DESCRIPTION

Fig. 1 is a block diagram of a system 10 for requesting content from a content server. In one implementation, the content may include advertisements ("ads"), and the content server can be an ad server. Different types of content can also be requested, e.g., weather, driving directions, trivia, etc.

In one implementation, a client system 100 is configured to visit web pages over a network, e.g., the Internet. The client system 100 can, for example, be a web browser, or a computing device executing network navigation software, etc. The web address (e.g., Uniform Resource Locator (URL)) visited by the client system 100 can be resolved to

identify a publisher 102, e.g. a server, hosting the corresponding web page. The client system 100 thus sends a web page content request 104 to the publisher 102 for the web page content 106. The publisher 102, in response to the request, provides the web page content 106 to the client system 100 as, e.g., an HTML document containing JavaScript. The web page content 106 can include one or more content presentations. In an implementation, the content presentations can include advertisement slots for advertisements to be served by an ad server. Other content presentations can also be used.

The web page content 106 provided by the publisher 102 includes a reference to a set of instructions 108. In an implementation, the instructions 108 include storing instructions 108a, timing instructions 108b and request instructions 108c that are used to render and present the requested content, e.g., advertisements. In an implementation, the instructions 108 are provided by a content server 134, e.g., and ad server, and are stored at the client system 100, such as in a cache associated with a web browser. In an implementation, the client system 100 can determine for each subsequent access to the stored instructions 108 whether the stored instructions 108 are the most recent version. If the stored instructions 108 are not the most recent version, the client system 100 can request the most recent version of instructions 110, which can include storing instructions 110a, timing instructions 110b and request instructions 110c from the content server 134. Upon receiving the most recent version of the instructions 110, the client system 100 stores the most recent version of the instructions 110 as the stored instructions 108.

The web page content 106 can define content slots 112 - 120 that are configured to display content from the content server 134. Though reference is made to a single content server 134, plural content servers can provide content to a single web page. In an implementation, the content slots 112 - 120 are ad slots that are defined within HTML tags. The instructions 108 generate content requests 122 - 130 that are issued to request content to fill the content slots 112 to 120. In an implementation, the requests 122 to 130 are stored in a data store 132, such as a buffer 132, and then sent to the content server 134 in one or more requests 136 and 140. The content server 134 processes the received individual or combined requests 136 and returns identified content 138 to the client system 100. The identified content 138 is then displayed as part of the publisher's web page in the corresponding content slots, e.g., content slots 112, 114 and 116.

When the client 100 requests content from the publisher 102, errors may be encountered because of mistakes in the tagging of the pages or in other environmental and

operational conditions. Such errors include syntax errors, semantic errors, operational errors and environmental errors. Syntax errors are encountered when a web page violates rules for associated with a given protocol, e.g., HTML syntax or JavaScript syntax. Many HTML editors will detect syntax errors. Some common syntax errors include mismatched HTML tags, for example a closing tag is missing, which would prevent the page content from displaying correctly. Incorrect JavaScript syntax, such as a missing quote in a JavaScript function call, is another example.

Semantic errors can also be protocol based. For example, semantic errors can occur when JavaScript function calls are missing or occur in the incorrect sequence. Semantic errors can arise when an invalid or incorrect parameter value name is specified in a protocol (e.g., JavaScript) function calls. Examples include, an incorrect slot name, out-of-sequence JavaScript tags, and a page that attempts to fill a slot that has not been added.

Operational errors occur when content is not displayed as expected because of a discrepancy between tags in the web page content 106 and the information stored on the content server 134. Common operational errors include latency issues within the content server 134 where changed values have not fully propagated through the content server 134, disabled slots in the web page content 106, and particular content not appearing due to frequency capping, day-parting, date or budget issues.

Environmental errors occur when web page content 106 is successfully retrieved, but the client system 100 (e.g., web browser) fails to display the web content 106. For example, the client system 100 may not have JavaScript enabled or the client system 100 may not be supported.

The above errors are provided for exemplary purposes, as many types of errors can occur during the processes of defining, requesting, serving and displaying of web page content 106.

With reference to Fig. 2, should one of the above exemplary errors occur, in accordance with some implementations, an example process 200 to provide debugging information begins with a request for a document or content from a web site where errors are encountered is made by including a flag (or other indicator) with the request (step 202). For example, a flag "google_debug" can be added to the URL of a web page content location as follows: http://www.website-with-errors.com/index.htm?google_debug.

Next, the requested document is received (step 204), and various conditions are tested to diagnose the cause of the errors (step 206). In some implementations, the content serving

(e.g., JavaScript) tags within the web page content 106 implement diagnostic logic. For example, a script file within the web page content 106 can test for various conditions, and upon the existence or absence of such conditions, write information to a debugging user interface. The scrip file diagnostic logic tests to determine error related to, but not limited to: an identifier of the publisher 102, an order in which the web page content is served, if a slot is defined or disabled, if a slot was previously defined, attributes of slots, if a requested URL is too long, if an iFrame should be created for a slot (i.e., a frame inserted within a Web page which is not bound to the side of a browser window), if an HTML DIV element should be created for a slot, and if content for a slot has been received.

In some implementations, the content serving (e.g., JavaScript) tags within the web page content 106 implement tracing logic. Such tracing logic can capture server-bound URLs and responses from the content server 134 which are displayed in the debugging user interface. Displaying missing parameters in the URL or escaping issues in the responses aids in debugging errors. In some implementations, other information such as the UserAgent is captured and displayed.

In some implementations, a timer is started as each URL or request is communicated the content server 134 or publisher 102, which times-out when a response is received by the client 100. The timing information can be used to determine where environmental and operational delays are encountered.

After the conditions are tested, a debugging user interface is spawned (step 208) and information regarding the results of the testing are written to the debugging user interface (step 210). In some implementations, the debugging user interface is created by JavaScript code that provides a separate browser window. A JavaScript class provides methods such as writeInfo, writeWarning, writeError to write various types of information to the debugging window. The methods also display timing information to show the length of time consumed by a particular operation, and color-code errors and unusual circumstances to identify problems.

Fig. 3 is a flow chart of another exemplary process 300 to debug errors in retrieving content from the content server 134. A document or content 106 from the publisher 102 is requested, where the request include an indicator (step 302). The document or content 106 is received by the client 100 (step 304). Content from the content server is then received (step 306). In some implementations, the content 106 received from the publisher 102 includes a

script having instructions that requests content from the content server 134 and provides that content to the client 100.

Next, the instructions to obtain content from the content server are debugged (step 308). In some implementations, the content serving (e.g., JavaScript) tags within the web page content 106 implement diagnostic and/or tracing logic, as described above with regard to Fig. 2. The results of the debugging are displayed (step 310). In some implementations, the debugging user interface is created by JavaScript code that provides a separate browser window, within which information is written, as noted above.

Below is an example portion of HTML code that requests content (e.g., ads) from the content server 134 and displays the web page content 106. In the example below, one slot is requested from the content server 134, and additional slots can be requested.

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
    <!-- download Google Ads JavaScript -->
    <script language="JavaScript" src="./google_service.js">
    </script>
    <script language="JavaScript">
        GS_googleAddAdSenseService("ca-pub1");
        GS_googleEnableAllServices();
    </script>
    <!-- JavaScript for slot TOPSLOT -->
    <script language="JavaScript">
        GA_googleAddSlot("ca-pub1", "TOPSLOT");
    </script>
    <!-- JavaScript for retrieving ads -->
    <script language="JavaScript">
        GA_googleFetchAds();
    </script>
</head>
<body onload="">
    <!-- create iframe for TOPSLOT -->
    <script language="JavaScript">
        GA_googleFillSlot("TOP2SLOT");
```

```
</script>
<div id="content">
<p> Sample page content.
</p>
</div>
</body>
</html>
```

In the example, a script file (google_service.js) is designated in the header for serving content (e.g., slots 112-120) that is displayed as part of the web page content 106. The publisher has added a particular slot "TOPSLOT;" however, the publisher has indicated that a slot "TOP2SLOT" is to be filled. The slot TOP2SLOT does not exist either because it is misspelled or missing. When the client system 100 executes the script, the diagnostic logic generates warnings using the writeError() method or errors using the writeError () method based on this mismatch. An example of this is shown in Fig 4, where the debugging user interface 400 created by the diagnostic logic indicates that the slot has not been defined.

Fig. 5 illustrates another example debugging user interface 500 where the publisher has not defined any slots and the script generates a warning. When the publisher attempts to render TOPSLOT, an error message is generated in the debugging user interface 500.

Fig. 6 illustrates a debugging user interface 600, showing other errors such as a missing publisher identifier, TOPSLOT has been defined, a missing slot name, duplicate definitions of TOPSLOT, and other transactional information.

This written description sets forth the best mode of the invention and provides examples to describe the invention and to enable a person of ordinary skill in the art to make and use the invention. This written description does not limit the invention to the precise terms set forth. Thus, while the invention has been described in detail with reference to the examples set forth above, those of ordinary skill in the art may effect alterations, modifications, and variations to the examples without departing from the scope of the invention.

What is claimed is:

1. A method, comprising:

requesting a document containing a script at a location, the request including an indicator;

receiving the document in response to the request;

executing the script to test a status of a condition in response to receipt of the indicator; and

displaying the status of the condition.

2. The method of claim 1, further comprising:

displaying a window; and

displaying the status in the window.

3. The method of claim 2, further comprising:

determining a status state; and

color-coding the status based the status state.

4. The method of claim 2, further comprising:

displaying timing information in the window to show a length of time required complete an operation specified by the condition.

5. The method of claim 2, further comprising:

displaying requests made to a content server in the window; and

display responses from the content server in the window.

6. The method of claim 1, wherein executing the script to test the status of the condition further comprises testing if function calls specified in the document are missing or occur in an incorrect sequence.

7. The method of claim 1, wherein executing the script to test the status of the condition further comprises determining if the document includes an incorrect slot name, out-of-sequence JavaScript tags, or attempts to fill a slot that has not been added.

8. The method of claim 1, wherein executing the script to test the status of the condition further comprises determining if a discrepancy exists between information specified by the document and information stored on a content server.

9. The method of claim 8, wherein the discrepancy comprises one of differences between tags specified in the document and information on the content server, latency issues caused by delays in the content server, or disabled slots in the document.

10. The method of claim 1, wherein executing the script to test the status of the condition further comprises testing if content is successfully received by but not displayed.

11. The method of claim 1, wherein the request is a Uniform Resource Locator (URL) of the document and the indicator is a flag added to the URL.

12. A system, comprising:

a content server configurable to receive content;

a publisher server operatively coupled to the content server and configurable to determine a context from the content; and

a content repository operatively coupled to the content server and configurable for providing the content server with content associated with a context,

wherein the publisher server provides computer executable instructions to a client device upon receiving a request from the client device, the computer executable instructions being executed by the client device to determine conditions related to the communication of content from the content repository to the client device.

13. A system, comprising:

a processor configurable to request content from a remote location, the content including computer executable instructions to determine debugging information associated with the request; and

an interface operatively coupled to the processor and configurable to display the debugging information, the debugging information including errors associated with the display of the content in the interface.

14. A computer-readable medium having instructions stored thereon, which, when executed by a processor, causes the processor to perform the operations of:

displaying first content provided by a publisher;

displaying second content provided by a content server;

debugging instructions to obtain the second content from the content server; and

displaying results of the debugging in an interface.

15. The computer-readable medium of claim 14, where displaying results of the debugging comprises:

determining a severity of the results; and

color-coding the results based the severity.

16. The computer-readable medium of claim 14, where displaying results of the debugging comprises displaying timing information in the interface to show a length of time required complete a predetermined instruction.

17. The computer-readable medium of claim 14, where debugging instructions comprises testing if function calls are missing or occur in the wrong sequence.

18. The computer-readable medium of claim 14, where debugging instructions comprises testing if the instructions specify one of an incorrect slot name, out-of-sequence JavaScript tags, or fills a slot that has not been added.

19. The computer-readable medium of claim 14, where debugging instructions comprises testing to determine if discrepancies exist between the instructions and information stored on the content server.

20. The computer-readable medium of claim 19, where the discrepancies include differences between tags specified in the instructions and information on the content server, latency issues caused by delays in the content server, or disabled slots specified by the instructions.

21. The computer-readable medium of claim 14, where debugging instructions comprises further comprises testing if content is successfully received but not displayed.

22. A system, comprising:

means for receiving a document provided by a remote publisher;

means for executing instructions contained in the document, where the instructions fetch content from a content server;

means for debugging the instructions to determine errors in the fetching of content from the content server; and

means for displaying the errors.

23. A method, comprising:

examining a request for a document to determine a presence of errors, the document including a script to debug requests for content; and

writing error conditions to a user interface presented to the content requestor.

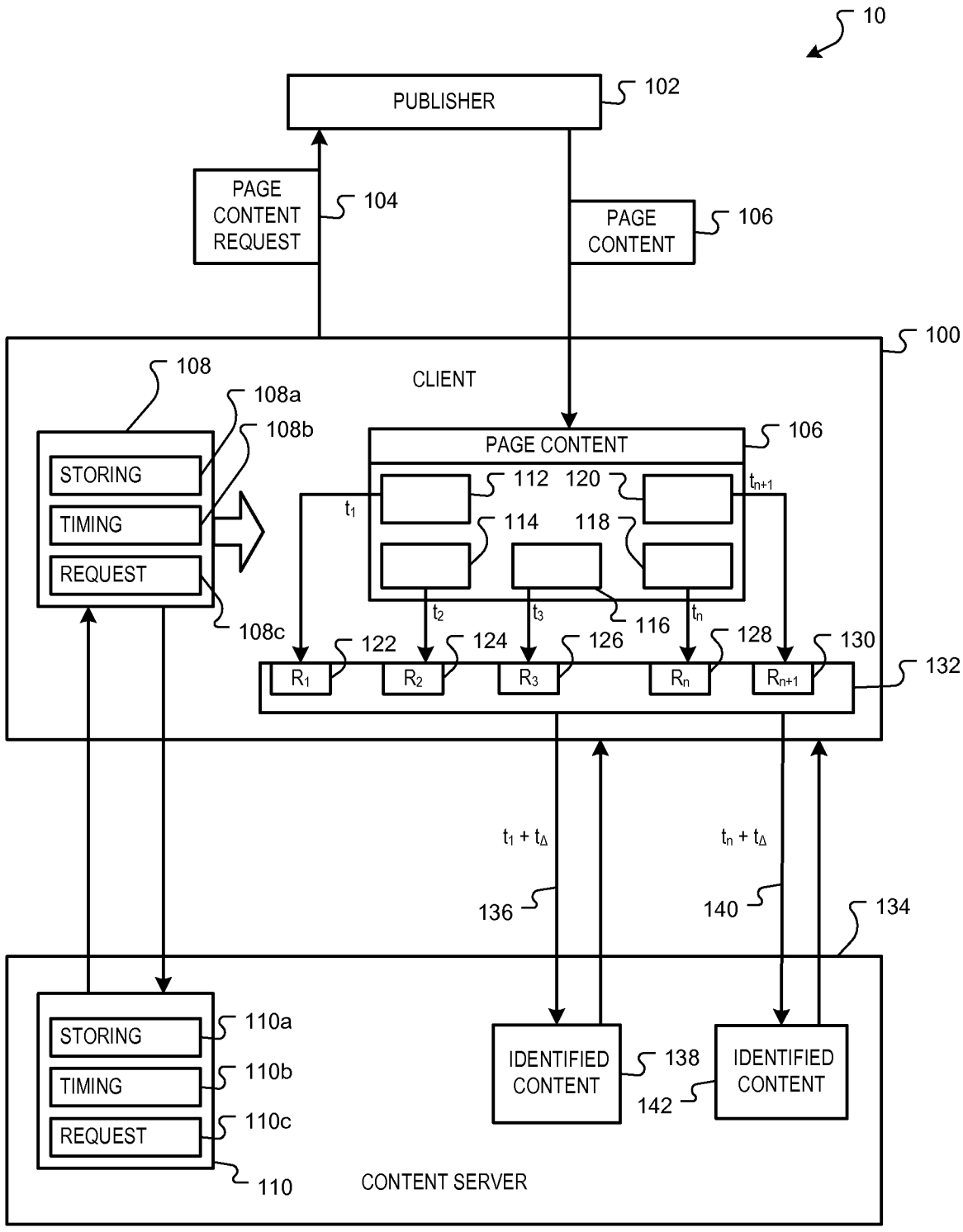

10
PUBLISHER
102
PAGE CONTENT REQUEST
104
PAGE CONTENT
106
100
CLIENT
108
108a
108b
STORING
TIMING
REQUEST
108c
PAGE CONTENT
106
112
120
114
118
116
t_1
t_2
t_3
t_n
t_{n+1}
122
124
126
128
130
132
R_1
R_2
R_3
R_n
R_{n+1}
$t_1 + t_\Delta$
$t_n + t_\Delta$
136
140
134
STORING
TIMING
REQUEST
110a
110b
110c
110
CONTENT SERVER
IDENTIFIED CONTENT
138
142
IDENTIFIED CONTENT

FIG. 1

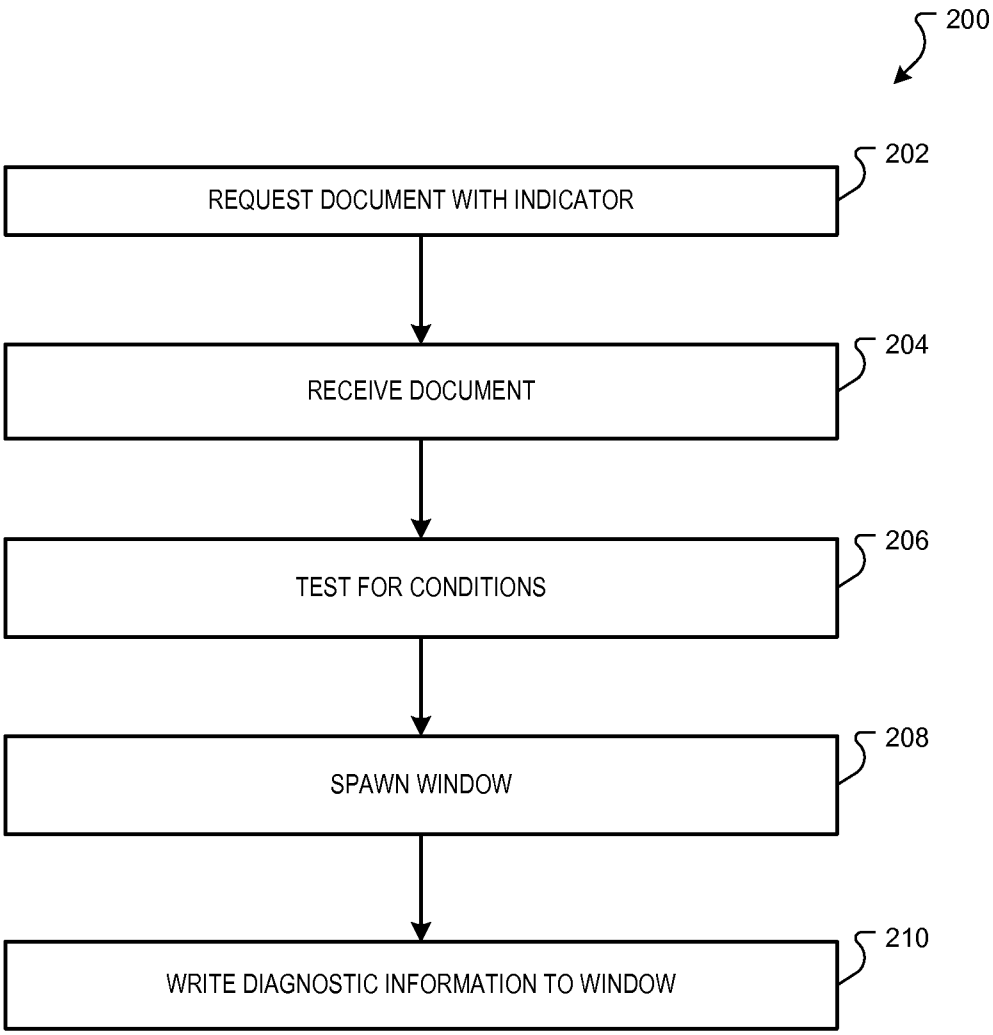

200
202
REQUEST DOCUMENT WITH INDICATOR
204
RECEIVE DOCUMENT
206
TEST FOR CONDITIONS
208
SPAWN WINDOW
210
WRITE DIAGNOSTIC INFORMATION TO WINDOW

FIG. 2

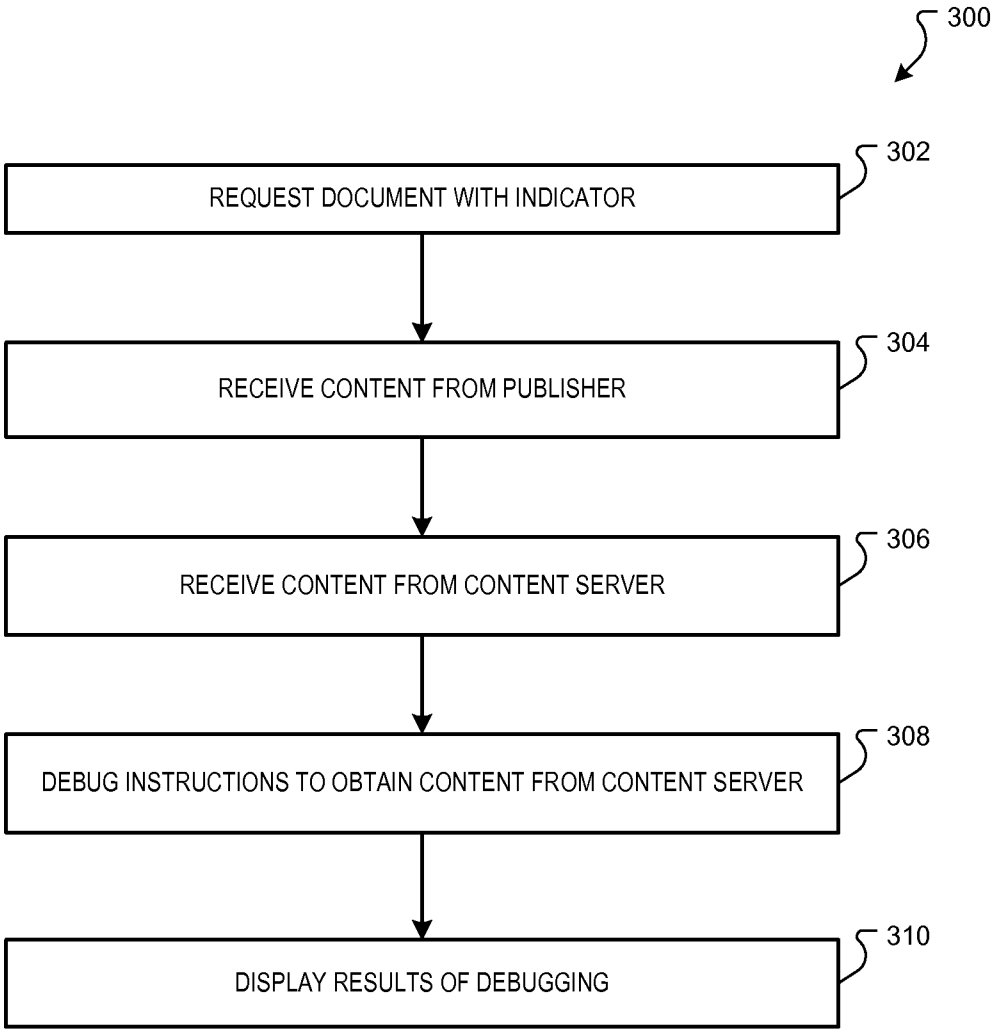

300
302
REQUEST DOCUMENT WITH INDICATOR
304
RECEIVE CONTENT FROM PUBLISHER
306
RECEIVE CONTENT FROM CONTENT SERVER
308
DEBUG INSTRUCTIONS TO OBTAIN CONTENT FROM CONTENT SERVER
310
DISPLAY RESULTS OF DEBUGGING

FIG. 3

400

Goggle Debug Window

Submit Data

Offset (msec)	Level	Message
0	Information	goggle_ads.js is being loaded at Monday, July 31, 2006 1:56:12PM
0	Information	publisher URL=http://www.corp.goggle.com/-stsun/ads2/badfill.html?google_debug
0	Information	user agent=Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5
0	Information	google_ads.js finished loading
0	Information	Slot publisher id=ca-pub1, name =TOPSLOT has been added
0	Information	[AdData:[GA_GoogleAdSlot: pubid=ca-pub1, name=TOPSLOT, loaded=0, tries=0]]
0	Information	Issuing fetch ad attr call with ./slotdata.js?client=ca-pub1
63	Information	Attributes received for slots [TOPSLOT: width=336, height=280, expandable=false, enabled=true]
63	Information	Using Multiple Call, Asynchronous Implementation
63	Information	[AdData:[GA_GoggleAdSlot: pubid=ca-pub1, name=TOPSLOT, loaded=0, tries=0]]
126	Information	FetchAd generated a correlator=11543793724 91
141	**Error**	**Skipping undefined Ad Slot("TOP2SLOT")**

FIG. 4

500
Goggle Debug Window
Submit Data
Offset (msec)
Level
Message
0 Information goggle_ads.js is being loaded at Monday, July 31, 2006 1:57:16PM
0 Information publisher URL=http://www.corp.google.com/-stsun/ads2/noslot.html?google_debug
0 Information user agent=Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5
0 Information google_ads.js finished loading
0 Information [AdData:]
0 Information Issuing fetch ad attr call with ./slotdata.js?client=ca-pub1
78 Information Attributes received for slots
78 Information Using Multiple Call, Asynchronous Implementation
78 Warning No slots defined on page
78 Error This slot has not ben defined: TOPSLOT

FIG. 5

Goggle Debug Window

600

Submit Data

Offset (msec)	Level	Message
0	Information	goggle_ads.js is being loaded at Monday, July 31, 2006 1:58:51PM
0	Information	publisher URL=http://www.corp.goggle.com/-stsun/ads2_error.html?google_debug
0	Information	user agent=Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.5) Gecko/20060719 Firefox/1.5.0.5
0	Information	google_ads.js finished loading
0	**Error**	**publisher id cannot be null or empty**
0	**Error**	**publisher id cannot be null or empty**
0	**Error**	**slot name cannot be null or empty for publisher ca-pub1**
0	**Error**	**slot name cannot be null or empty for publisher ca-pub1**
15	Information	Slot publisher id=ca-pub1, name=TOPSLOT has been added
15	**Error**	**Slot TOPSLOT has already been defined**
15	Information	[AdData:[GA_GoggleAdSlot: pubid=ca-pub1, name=TOPSLOT, loaded=0, tries=0]]
15	Information	Issuing fetch ad attr call with ./slotdata js?client=ca-pub1
62	Information	Attributes received for slots [TOPSLOT: width=336, height=280, expandable=false, enabled=true]
62	Information	Using Multiple Call, Asynchronous Implementation
62	Information	[AdData:[GA_GoggleAdSlot:pubid=ca-pub1, name=TOPSLOT, loaded=0, tries=0]]
78	Information	FetchAD generated a correlator=1154379531347
78	Information	Generated iframe for slot TOPSLOT with <iframe id=goggle_ads_ iframe_TOPSLOT width=”336" height=”280" vspace=”0" hspace=”0" src=”./data.js?&client=ca-pub1& slotname=TOPSLOT&ref=http%3A%2F%2Fwww.corp.goggle.com%2F-stsun%2Fads2%2F&url=http%3A%2F% 2Fwww.corp.goggle.com

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2007/083048

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/22(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 8 : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

PAJ, FPD, USPAT, eKIPASS, IEEE, YAHOO, GOOGLE , Keyword: "script, debugging, web"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,668,369 B1 (WILLIAM H. KREBS et al) 23 December 2003 (See Abstract, Fig.6, column 2 line 54 - column 3 line 2, column 4 line 28-column 5 line 7)	1-23
A	US 6,353,923 B1 (PHILLIP LEE BOGLE et al) 05 March 2002 (See Abstract, Figs.2-7, column 3 line 17- column 4 line34, column 7 line60-column 14 line 39)	1-23
A	US 6,463,578 B1 (DAVID PHILLIP JOHNSON) 08 October 2002 (See Abstract, Figs.4-5, column 2 line 13-59, column 6 line 25-column 7 line 43)	1-23
A	JP 2004-185063 A (HITACHI INFORMATION SYS LTD) 02 July 2004 (See Abstract, paragraphs [0008]-[00200)	1-23
A	EP 1420562 A2 (MICROSOFT CORP) 19 May 2004 (See Abstract, paragraphs [0008],[0042]-[0071])	1-23

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 FEBRUARY 2008 (22.02.2008)	**22 FEBRUARY 2008 (22.02.2008)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

Sun Dong Guk

Telephone No. 82-42-481-8248

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2007/083048

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US06668369B1	23.12.2003	NONE	
US06353923B1	05.03.2002	US2001005852AA	28.06.2001
		US6275868B1	14.08.2001
		US7203926BB	10.04.2007
US06463578B1	08.10.2002	NONE	
JP2004185063A	02.07.2004	NONE	
EP1420562A2	19.05.2004	EP1420562A3	07.06.2006
		JP2004164617A2	10.06.2004