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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 September 2005 (09.09.2005)

PCT

(10) International Publication Number
WO 2005/083967 A1

(51) International Patent Classification[7]: H04L 29/06, 29/08

(21) International Application Number: PCT/EP2005/050299

(22) International Filing Date: 24 January 2005 (24.01.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/789,573 27 February 2004 (27.02.2004) US

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41 North Harbour, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BURCKART, Erik** [US/US]; 2348 Persimmon Ridge Drive, Raleigh, NC 27604 (US). **FONTES, Stephen** [US/US]; 1336 Winter Walk Circle, Morrisville, NC 27560 (US). **LANZEN, Craig** [US/US]; 3157 Lennox Ct, Lambertville, MI 48144 (US).

(74) Agent: **LITHERLAND, David, Peter**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

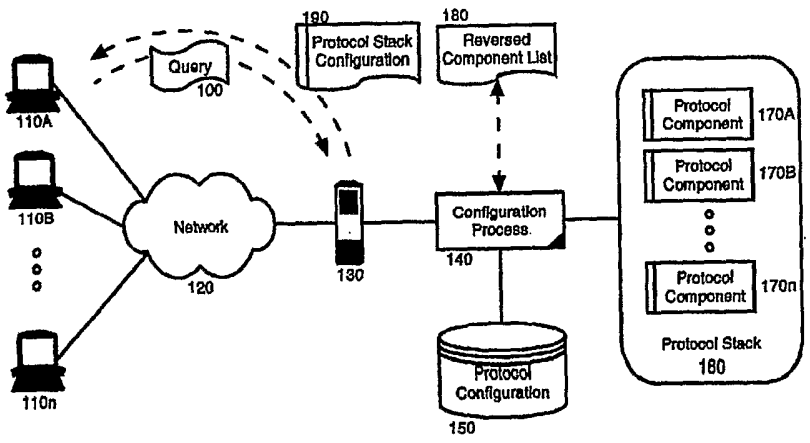
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR,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

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SERVER-SIDE PROTOCOL CONFIGURATION OF ACCESSING CLIENTS



(57) Abstract: Disclosed is a method and system for configuring a client-side communications protocol stack. In one method, a mapping can be consulted to determine a set of client-side protocol stack components which correspond to a set of protocol stack components in a server side protocol stack instance. Subsequently, a listing can be created of the determined set of client-side protocol stack components. Finally, the listing can be published for access by externally disposed client computing processes. Preferably, the listing can be reversed. Also, the consulting step can further include determining at least one attribute to be applied to at least one of the client-side protocol stack components when enabling a client-side protocol stack to interoperate with the server-side protocol stack instance.

WO 2005/083967 A1

Description

SERVER-SIDE PROTOCOL CONFIGURATION OF ACCESSING CLIENTS

Technical Field

[001] The present invention relates to the field of inter-process communications and more particularly to configuring client processes for communication with server processes.

Background Art

[002] Inter-process communications relates to the exchange of information between two or more computing processes, whether the processes are disposed together within a single, computing device, or whether the processes are remotely positioned from one another about a computer communications network. Traditionally, separate computing processes can communicate with one another over communications links established for the purpose of exchanging information. In most circumstances, information can be exchanged using a common communications protocol negotiated prior to the establishment of, or during the course of establishing a communication link.

[003] In many cases, the common communications protocol utilized by two or more computing processes to exchange information can be predetermined before any attempt is made to establish a communication link. For example, in the context of the Internet, Web content can be exchanged between a content server and a content browser using the hypertext transfer protocol (HTTP) over TCP/IP. Still, communications protocols are not necessarily limited to the physical network layer based exchange of data. Rather, communications protocols can include any collection of operations and communication configurations required to permit the exchange of information between two or more computing processes. In this regard, a communications protocol can include not only the mechanism for transferring data between the processes, but also ancillary settings and operations such as encryption, compression and tunneling protocols, to name a few.

[004] In certain situations, a communications protocol will not have been pre-determined as between a client computing process and a server computing process. In particular, where the server computing process has been configured to communicate with a multiplicity of client computing processes using many different communications protocols, it cannot be pre-determined which computing process to utilize for any one session between client and server. In consequence, oftentimes, client computing processes are left to select a suitable communications protocol through a process of trial and error.

Specifically, in the prototypical circumstance, the client process can include a table of multiple possible communications protocols. As such, the client process can attempt to establish a communication link with the server process using each possible communications protocol until such time as a communication link has been successfully established.

[005] The foregoing process for establishing a communication link between server processes can be effective in as much as ultimately, a communication link can be established using one of the known communications protocols. Still, the skilled artisan will recognize the terrible waste of resources which can occur with each failure to establish a communication link using a particular communications protocol other than the communications protocol ultimately selected in establishing the communication link. To the extent possible, it would be preferable to avoid the process of trial and error in selecting a communications protocol to achieve a greater level of performance.

Disclosure of Invention

[006] The present invention seeks to address the deficiencies of the art in respect to inter-process communications and provides a method and system for configuring a communications protocol stack in a client side computing process. In one preferred aspect of the invention, an inter-process communications protocol configuration system can include a mapping of server-side protocol stack components to client-side protocol stack components in a server-side protocol stack. The system also can include a listing of dependent ones of the server-side protocol stack components in the server-side protocol stack based upon the mapping. Finally, the system can include a publishable interface to the listing.

[007] The mapping can include a set of linkages between protocol stack components of the server-side protocol stack and corresponding protocol stack components required to exist in an associated client-side protocol stack. The mapping also can include a set of attributes for selected ones of the corresponding protocol stack components. In a preferred aspect of the invention, the listing can be a reverse listing having a reverse ordered list of the dependent ones of the server-side protocol stack components in the server-side protocol stack based upon the mapping. Also, it is preferred that the publishable interface be a native directory interface to at least one object incorporating the listing.

[008] In a method for configuring a client-side communications protocol stack, a mapping can be consulted to determine a set of client-side protocol stack components which correspond to a set of protocol stack components in a server side protocol stack

instance. Subsequently, a listing can be created of the determined set of client-side protocol stack components. Finally, the listing can be published for access by externally disposed client computing processes. Preferably, the listing can be reversed. Also, the consulting step further can include determining at least one attribute to be applied to at least one of the client-side protocol stack components when enabling a client-side protocol stack to interoperate with the server-side protocol stack instance.

[009] Conversely, a method for configuring a client-side communications protocol stack can include accessing a listing of required client-side protocol stack components which correspond to protocol stack components aggregated in a server-side protocol stack instance. It can be determined from the listing whether a client-side protocol stack instance already exists which includes the required client-side protocol stack components. If the client-side protocol stack instance already exists, the client-side protocol stack instance can be re-used to communicate with the server-side protocol stack. Otherwise a new instance of a client-side protocol stack can be created using the required client-side protocol stack components.

Brief Description of the Drawings

[010] An embodiment of the present invention will now be described, by way of example only, with reference to the following drawings in which:

[011] Figure 1 is a schematic illustration of a computing system configured for the server-side configuration of accessing clients in accordance with an embodiment of the present invention;

[012] Figure 2 is a flow chart illustrating a process for linking corresponding client and server protocol elements in a server side configuration in the system of Figure 1;

[013] Figure 3 is a flow chart illustrating a process for starting a protocol stack of different protocol elements in a server in the system of Figure 1; and,

[014] Figure 4 is a flow chart illustrating a process for selecting and configuring a specific protocol stack of different client side protocol elements based upon a server specified protocol stack in the system of Figure 1.

Mode for the Invention

[015] The present invention is a method, system and apparatus for intelligently configuring a client computing process for communication with a server computing process based upon a dynamically selected and configured communications protocol stack. In accordance with the present invention, different corresponding protocol elements or components in a protocol stack can be mapped as between the server

computing process and one or more client computing processes configured to access the server computing process. For server side protocol components having specifically defined attributes which map to attributes in a prospective client side protocol component, those attributes can be specified in the mapping as well. The mapping along with the specified attributes can be stored for subsequent use when deploying a protocol stack in the server.

[016] When a protocol stack is to be deployed for use in the server computing process, a list of dependent protocol components for the protocol stack can be generated and the list can be reversed in terms of order. For each of the protocol components in the list of dependent protocol components, the values of any properties can be retrieved. Consequently, a client configuration for the protocol stack can be created using both the reversed list of dependent protocol components and the retrieved values of any properties for the property dependencies of the protocol stack. Subsequently, the configuration can be published so that client processes attempting to communicate with the server can discover the stored configuration.

[017] When a client computing process attempts to communicate with the server computing process, the client computing process first can query the server specified protocol configuration through its prior publication. Once discovered, the client process can determine whether the specified protocol configuration already exists in the client process. If so, the protocol configuration can be re-used to communicate with the server process. Otherwise, the specified configuration can be created in the server and configured with the associated property values for the dependencies of the configuration. When the protocol stack becomes operable, the client process can use the protocol stack to communicate with the server process.

[018] In more particular illustration of the foregoing, Figure 1 is a schematic illustration of an exemplary albeit non-exclusive computing system configured for the server-side configuration of accessing client processes in accordance with the inventive arrangements. The system can include one or more client computing process 110A, 110B, 100n programmed for communication coupling to a server process 130 over a computer communications network 120. Notably, each client computing process 110A, 110B, 110n can communicate with the server computing process 130 using an established communications protocol. The established communications protocol can include not only a physical and applications protocol, for example HTTP using TCP/IP, but also the operation of one or more ancillary procedures such as an encryption or compression process.

[019] The server communications process 130 can be coupled to a configuration processor 140. The configuration processor 140 can establish a set of protocol configurations 150. Specifically, the configuration processor 140 can map individual components required for the operation of the protocol in the server computing process 130 with corresponding required components in any one of the client computing processes 110A, 110B, 110n. When a protocol stack is to be created in the server computing process 130, the component dependencies of the set of components in the stack can be listed and subsequently reversed to produce a reverse configuration 180. Individual property values of the components in the reverse configuration 180 further can be determined and the combination can be published as a protocol stack configuration 190. In particular, as it is well known in the art, the configuration of the protocol stack 190 can be published in the global names space as an interface to one or a set of class objects whose contents can be recognized through a native directory interface.

[020] In operation, when one of the client computing processes 110A, 110B, 110n desires to establish a communication link with the server computing process 130, the client computing process 110A, 110B, 110n first can forward a query 100 to the server computing process 130 requesting that the server computing process 130 identify the preferred communications protocol configuration. In response, the server computing process 130 can return access to the protocol stack configuration 190 with which the requesting one of the client computing processes 110A, 110B, 110n can select and configure the preferred communications protocol stack as defined by the protocol stack configuration 190.

[021] The process of configuring a client computing process to communicate with the server computing process using a preferred communications protocol configuration can include linking server side and client side protocols to one another in a mapping. The process also can include starting a communications protocol stack in the server computing process and developing a corresponding publishable client side communications protocol stack configuration based upon the mapping. Finally, the process can include selecting and configuring a specific protocol stack of different client side protocol components based upon a server specified protocol stack.

[022] To further illustrate the foregoing process, Figure 2 is a flow chart illustrating a process for linking corresponding client and server protocol elements in a server side configuration in the system of Figure 1. Beginning in block 210, a first protocol set can be loaded for configuration. In block 220, each of the protocol components in the set

for the server computing process can be linked or mapped to corresponding dependent protocol components for a prospective communicating client process. For example, the use of the HTTP application level protocol in the server computing process can be mapped to HTTP in the client computing process. Likewise, TCP as a physical control layer protocol in the server computing process can be mapped to TCP in the client computing process. As yet another example, a decompression process in the server computing process can be mapped to a compression process in the client computing process.

[023] In any event, in decision block 230, it can be determined whether specific attributes of selected ones of the dependent protocol components are to be applied based upon the application of specific attributes corresponding ones of the protocol components in the server computing process. If attributes are to be mapped, the attributes can be so specified in the mapping in block 240. In either circumstance, in decision block 250, it can be determined whether additional protocol sets remain to be processed. If so, in block 260 the set of protocols can be loaded for configuration and the process can repeat in blocks 220 through 260. When no more protocol sets remain to be configured in the mapping, the process can end in block 270.

[024] Once the different protocol stack components and corresponding attributes have been linked in a mapping, a published configuration for a particular instance of a protocol stack can be produced using the mapping. More particularly, Figure 3 is a flow chart illustrating a process for starting a protocol stack of different protocol elements in a server in the system of Figure 1. Beginning in block 310, the protocol stack can be started. In an object-oriented environment, for example, the different protocol components aggregated into the protocol stack can be instantiated within a process address space and rendered operable therein.

[025] In block 320, a list of client side dependent components can be created by traversing each component in the server-side protocol stack, for instance, and identifying component dependencies from the mapping. Once the list of dependent components in the protocol stack has been created, in block 330, the ordering of the list can be reversed. In this way, a client computing process building a protocol stack based upon the list can be assured that each component in the stack can rely upon its dependencies. In any case, in decision block 340, it can be determined whether the dependencies of the stack are proper in that the dependencies rely upon an available component instance. If not, the process can end in block 350. Otherwise, the process can continue through block 360.

[026] In block 360, if proper dependencies exist, the values of the properties of each communications protocol component can be retrieved. Also, in block 370, a configuration for the protocol stack to be deployed in the client can be created based upon the reverse ordered list. Finally, in block 380, the retrieved property values can be attached to the created configuration for the client protocol stack. In block 390, on request the configuration for the client protocol stack can be passed to requesting client processes through any known mechanism. In a preferred aspect of the invention, the configuration can be published in the global namespace through a native directory interface.

[027] Once published the client side protocol stack configuration can be accessed by client computing processes interested in establishing a communication link with the server computing process using a communications protocol preferred by the server computing process. Importantly, the client computing process can select a suitable communications protocol without undertaking trial-and-error in guessing which protocol to use, and without having a pre-configured understanding of which protocol to use. More specifically, Figure 4 is a flow chart illustrating a process for selecting and configuring a specific protocol stack of different client side protocol elements based upon a server specified protocol stack in the system of Figure 1.

[028] Beginning in block 401, the preferred protocol stack configuration can be retrieved from the server computing process. In this regard, the client computing process can query the server computing process for the preferred protocol stack configuration, for instance by accessing the interface to the preferred protocol stack through a native directory interface to the preferred protocol stack. In decision block 420, it can be determined whether the preferred protocol stack is similar to an already existing instance of a protocol stack in the client computing process. If not, a new protocol stack instance can be created having each component specified within the configuration in proper dependency order. Additionally, the attributes of the components can be set as described in the preferred protocol stack and the property values for the properties of the components further can be set according to the preferred protocol stack.

[029] In decision block 420, however, if the preferred protocol stack is similar to an already existing instance of a protocol stack in the client computing process, it further can be determined in block 430 whether the existing instance of the protocol stack includes the same attributes specified by the preferred protocol stack. If not, as before in block 450 a new instance of the preferred communications protocol stack can be

created in the client. Otherwise, in block 440 the existing instance of the communications protocol stack can be re-used for communications with the server computing process.

[030] The present invention can be realized in hardware, software, or a combination of hardware and software. An implementation of the method and system of the present invention can be realized in a centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system, or other apparatus adapted for carrying out the methods described herein, is suited to perform the functions described herein.

[031] A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which, when loaded in a computer system is able to carry out these methods.

[032] Computer program or application in the present context means any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form. Significantly, this invention can be embodied in other specific forms and accordingly, reference should be had to the following claims, rather than to the foregoing specification, as indicating the scope of the invention.

Claims

[001] A method for configuring a client-side communications protocol stack, the method comprising the steps of: consulting a mapping to determine a set of client-side protocol stack components which correspond to a set of protocol stack components in a server side protocol stack instance; creating a listing of said determined set of client-side protocol stack components; and, publishing said listing for access by externally disposed client computing processes.

[002] The method of claim 1, wherein said consulting step further comprises the step of determining at least one attribute to be applied to at least one of said client-side protocol stack components when enabling a client-side protocol stack to interoperate with said server-side protocol stack instance.

[003] The method of claim 1, wherein said creating step further comprises the step of reversing an order of said created listing.

[004] A method for configuring a client-side communications protocol stack, the method comprising the steps of: accessing a listing of required client-side protocol stack components which correspond to protocol stack components aggregated in a server-side protocol stack instance; determining from said listing whether a client-side protocol stack instance already exists which includes said required client-side protocol stack components; and, if said client-side protocol stack instance already exists, re-using said client-side protocol stack instance to communicate with said server-side protocol stack, and otherwise creating a new instance of a client-side protocol stack using said required client-side protocol stack components.

[005] The method of claim 4, wherein said accessing step comprises the step of accessing said listing through a native directory interface.

[006] The method of claim 4, further comprising the steps of: identifying attributes to be applied to said required client-side protocol stack components; and, applying said identified attributes to said client-side protocol stack components in said new instance.

[007] The method of claim 6, wherein said re-using step comprises the steps of: re-using said client-side protocol stack instance to communicate with said server-side protocol stack only if said client-side protocol stack instance includes said identified attributes; and, otherwise creating a new instance of a client-side protocol stack using said required client-side protocol stack components and

including said identified attributes.

[008] An inter-process communications protocol configuration system comprising: means for mapping server-side protocol stack components to client-side protocol stack components in a server-side protocol stack; means for listing dependent ones of said server-side protocol stack components in said server-side protocol stack based upon said mapping; and, means providing a publishable interface to said listing.

[009] The system of claim 8, wherein said means for mapping comprises: a set of linkages between protocol stack components of said server-side protocol stack and corresponding protocol stack components required to exist in an associated client-side protocol stack; and, a set of attributes for selected ones of said corresponding protocol stack components.

[010] The system of claim 8, wherein said listing is a reverse listing comprising a reverse ordered list of said dependent ones of said server-side protocol stack components in said server-side protocol stack based upon said mapping.

[011] The system of claim 8, wherein said publishable interface is a native directory interface to at least one object incorporating said listing.

[012] A machine readable storage having stored thereon a computer program for configuring a client-side communications protocol stack, the computer program comprising a set of machine readable instructions which when executed by a machine method cause the machine to perform the steps of any of claims 1 to 7.

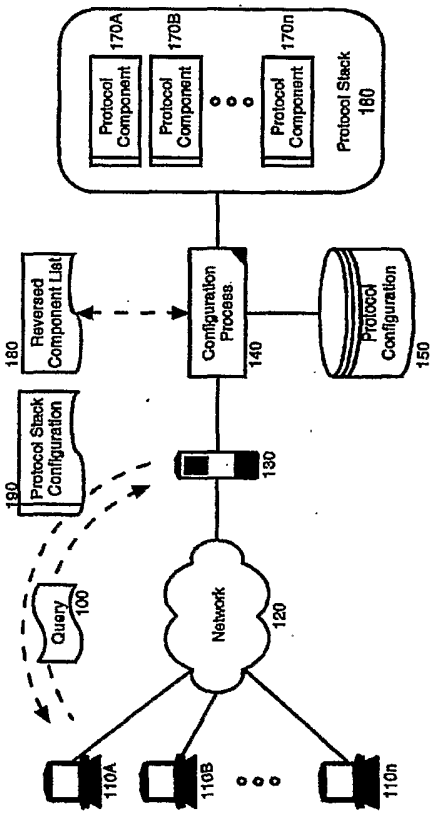

170A
170B
170n
Protocol Component
Protocol Component
Protocol Component
Protocol Stack 160
180
Reversed Component List
Configuration Process. 140
Protocol Configuration 150
190
Protocol Stack Configuration
130
Query 100
Network 120
110A
110B
110n

FIG. 1

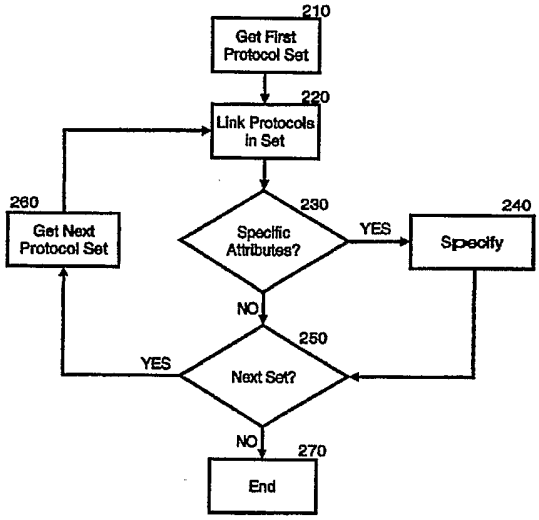

210
Get First Protocol Set
220
Link Protocols in Set
230
Specific Attributes?
YES
240
Specify
NO
260
Get Next Protocol Set
250
Next Set?
YES
NO
270
End

FIG. 2

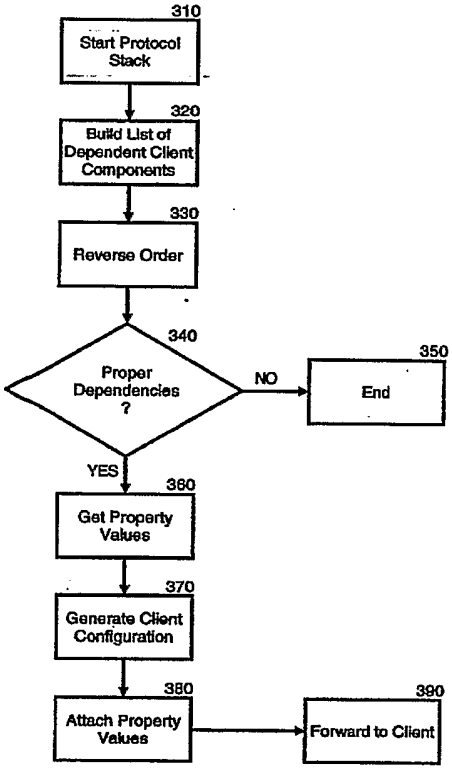

310
Start Protocol Stack
320
Build List of Dependent Client Components
330
Reverse Order
340
Proper Dependencies ?
NO
350
End
YES
360
Get Property Values
370
Generate Client Configuration
380
Attach Property Values
390
Forward to Client

FIG. 3

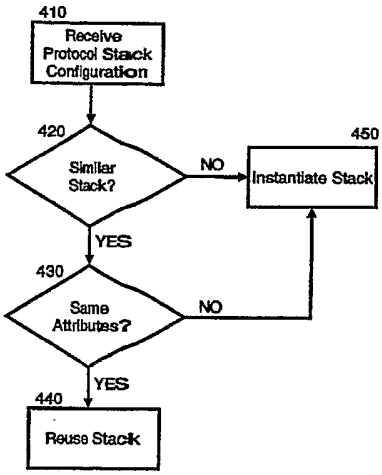

410
Receive Protocol Stack Configuration
420
Similar Stack?
NO
450
Instantiate Stack
YES
430
Same Attributes?
NO
YES
440
Reuse Stack

FIG. 4

INTERNATIONAL SEARCH REPORT

PCT/EP2005/050299

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L29/08

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 7 H04L

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

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

EPO-Internal, PAJ, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 6 212 160 B1 (BARBIERI RAYMOND A ET AL) 3 April 2001 (2001-04-03) abstract column 2, line 34 - column 3, line 41 figures 1,2	1-3,8-12 4-7
X	US 5 951 639 A (MACINNIS ET AL) 14 September 1999 (1999-09-14) abstract columns 3-4 columns 7-8 figures 2,3a,4,5	1-12
Y A	EP 1 292 160 A (TELEFONAKTIEBOLAGET LM ERICSSON) 12 March 2003 (2003-03-12) abstract paragraphs '0014! - '0016!, '0038! figures 1,2	4-7 1-3,8-12

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in 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 document 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 another 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
31 May 2005	07/06/2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Goya, J

INTERNATIONAL SEARCH REPORT

PCT/EP2005/050299

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 1 370 044 A (MICROSOFT CORPORATION) 10 December 2003 (2003-12-10) abstract paragraphs '0010! - '0015!, '0020!, '0027!, '0030!, '0031!, '0035! - '0037! figures 2,3	1-12

INTERNATIONAL SEARCH REPORT

PCT/EP2005/050299

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6212160 B1	03-04-2001	NONE	
US 5951639 A	14-09-1999	AU 1693597 A	02-09-1997
		DE 69712320 D1	06-06-2002
		DE 69712320 T2	07-11-2002
		EP 0880857 A1	02-12-1998
		WO 9730549 A1	21-08-1997
		US 2003028899 A1	06-02-2003
		US 6487723 B1	26-11-2002
EP 1292160 A	12-03-2003	EP 1292160 A1	12-03-2003
		WO 03024138 A1	20-03-2003
EP 1370044 A	10-12-2003	US 2004204071 A1	14-10-2004
		CN 1455570 A	12-11-2003
		EP 1370044 A2	10-12-2003
		JP 2004135258 A	30-04-2004