PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :		(11) International Publication Number:
G06F 9/46	A2	WO 00/34863
		(43) International Publication Date: 15 June 2000 (15.06.00)

(21) International Application Number: PCT/US99/25782

(22) International Filing Date: 2 November 1999 (02.11.99)

(30) Priority Data:
09/205,210 4 December 1998 (04.12.98) US

(71) Applicant: HONEYWELL INC. [US/US]; Honeywell Plaza, Minneapolis, MN 55408 (US).

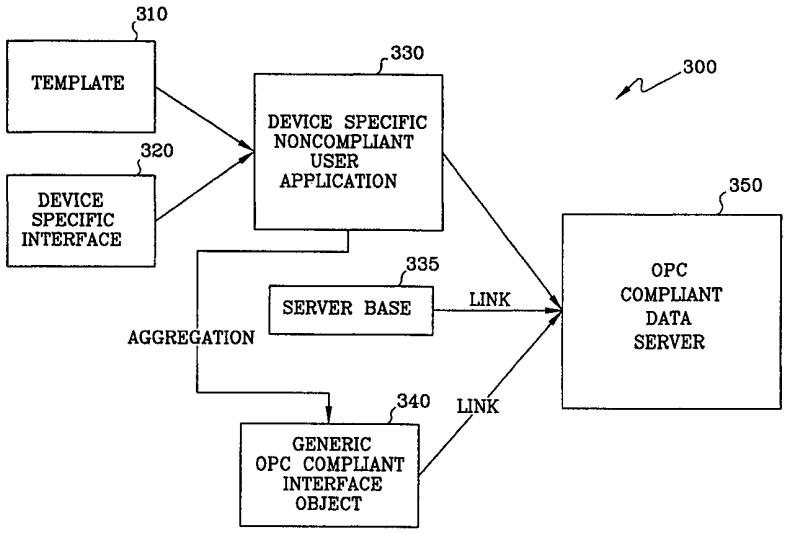
(72) Inventors: HAWKINSON, Ellen, B.; 3539 East Kachina Drive, Phoenix, AZ 85044 (US). KAAKANI, Ziad, M.; 5454 East Justin Road, Scottsdale, AZ 85254 (US). THOMAS, Christian, R.; 9246 East Hualapai Drive, Scottsdale, AZ 85255 (US). WEELDREYER, James, A.; 10205 North 78th Way, Scottsdale, AZ 85258 (US).

(74) Agent: MIOLOGOS, Anthony; Honeywell Inc., MN12–8251, Honeywell Plaza, P.O. Box 524, Minneapolis, MN 55440–0524 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SYSTEM AND METHOD FOR CONSTRUCTING AN OLE PROCESS CONTROL COMPLIANT DATA SERVER FROM A NONCOMPLIANT USER APPLICATION



(57) Abstract

A system for, and method of, constructing an Object Linking and Embedding (OLE) for Process Control (OPC) compliant data server from a device–specific noncompliant user application and a real–time process control system employing the system or the method. In one embodiment, the system includes: (1) a generic OPC compliant interface object and (2) a template associated with the generic OPC compliant interface object. The template is modifiable and combinable with a device–specific interface portion to yield the device–specific noncompliant user application. The device–specific noncompliant user application is dynamically linkable with and aggregates to the generic OPC compliant interface object to yield the OPC compliant data server.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD FOR CONSTRUCTING AN OLE PROCESS CONTROL COMPLIANT DATA SERVER FROM A NONCOMPLIANT USER APPLICATION

TECHNICAL FIELD OF THE INVENTION

The present invention is directed, in general, to real-time process control systems and, more specifically, to a system and method for constructing an Object Linking and Embedding (OLE) for Process Control (OPC) compliant data server from a noncompliant user application and a real-time process control system employing the system or method.

BACKGROUND OF THE INVENTION

Real-time process control systems were first implemented within a single computer system. As the need to monitor and control more physical devices increased, the complexity and size of the process control systems also increased. Shortly thereafter, single computer real-time process control systems were unable to process all the information within a timely manner as required by the real-time process control environments.

To correct this processing problem, real-time process control systems evolved into multiple computer systems that were connected to each other with proprietary communication interfaces. The multiple computer systems processed data locally and then communicated the information to the other computer systems over the proprietary communication interfaces. Since, the computer systems did not use a standard communication interface nor a standard protocol between each of the computer systems, modifications and additions to the systems were difficult, if not impossible.

This inter-computer incompatibility problem was resolved when the computer industry developed standardized networks and network protocols. Two of the industry standards were Ethernet and Transmission Control Protocol/Internet Protocol ("TCP/IP") used on 10 base 2 coaxial cable. Ethernet and TCP/IP allowed various computer systems the ability to communicate to each other without using proprietary communication interfaces.

The next evolution in real-time process control systems was object oriented distributed processing. In object oriented distributed processing, requesting programs

("clients") would call resource programs ("objects") to process a request. In this design, the clients and objects may be located on different computers on the network or are on the same computer. To facilitate a standardized way for clients to locate and communicate with objects, Microsoft Corporation developed the Component Object Model ("COM") protocol. The COM protocol, incorporated into software libraries called "COM libraries," defines a standardized interface for locating and communicating to objects over the network without requiring the clients to know the location of the desired objects.

The process control industry incorporated the COM standard and Object Linking and Embedding ("OLE") in its real-time process control standard, calling the resulting standard OLE for Process Control ("OPC"). The OPC standard defined the interfaces for distributed real-time process control object processing.

For companies to sell real-time process control systems, the real-time process control systems had to be COM and OPC compliant. This required the developers of the systems to have learned the intricacies of the COM and the OPC standards. Next, the programmers had to write detailed software programs that implemented the COM standards. Then, the programmers had to write detailed software programs that implemented the OPC standards which incorporated the COM standards. Finally, the programmers had to repeat this process for each type of distributed object processing that was performed within the real-time process control system.

When the programmers found an incorrect implementation of the COM and the OPC standards, the programmers had to correct each software program that was affected. This whole process incurred a great amount of time and expense. What is needed in the art is a way to rapidly develop COM/OPC compliant software for real-time process control systems.

SUMMARY OF THE INVENTION

To address the above-discussed deficiencies of the prior art, the present invention provides a system for, and method of, constructing an OPC compliant data server from a device-specific noncompliant user application and a real-time process control system employing the system or the method. In one embodiment, the system includes: (1) a generic OPC compliant interface object and (2) a template associated

with the generic OPC compliant interface object. The template is modifiable and combinable with a device-specific interface portion to yield the device-specific noncompliant user application. The device-specific noncompliant user application is dynamically linkable with and aggregates with the generic OPC compliant interface object to yield the OPC compliant data server.

The present invention therefore introduces the broad concept of dividing an OPC compliant data server into two distinct portions: a first portion that is generic and OPC compliant and a second portion that is specific to an underlying device or process control system and does not need to be entirely OPC compliant. A user wishing to create an OPC compliant data server is merely required to customize a template to yield the second portion. During run-time, the first and second portions are dynamically linked. The second portion aggregates the first, allowing the noncompliant second portion to inherit at least some of the attributes of the compliant first portion.

In one embodiment of the present invention, the system further includes a class factory that creates multiple related instances of objects. In a related embodiment, the class factory creates objects and manages global resources with respect to the objects. Those skilled in the pertinent art are familiar with class factories and their use in other object-oriented environments. The present invention can employ class factories to instantiate the objects needed to enable a particular server.

In one embodiment of the present invention, the system further includes a cache memory associated with the generic OPC compliant interface object. The generic OPC compliant interface object retrieves items from the cache memory in response to requests from multiple OPC clients. In a related embodiment, the generic OPC compliant interface object retrieves information from a device through the device-specific noncompliant user application and stores the information in the cache memory.

In one embodiment of the present invention, the class factory is capable of self-registration. However, the present invention need not be capable of self-registration.

In one embodiment of the present invention, the generic OPC compliant interface object is Component Object Module (COM) compliant. Those skilled in the pertinent art will perceive, however, that the present invention may be employed to advantage in object-oriented environments that are not COM-compliant.

The foregoing has outlined, rather broadly, preferred and alternative features of

the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art should appreciate that they can readily use the disclosed conception and specific embodiment as a basis for designing or modifying other structures for carrying out the same purposes of the present invention. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the invention in its broadest form.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

FIGURE 1 illustrates a block diagram of a real-time process control system that forms one environment within which the present invention can operate;

FIGURE 2 illustrates a block diagram of a real-time process control software architecture;

FIGURE 3 illustrates a block diagram of the overall process of creating an OPC compliant data server according to the principles of the present invention; and

FIGURE 4 illustrates a block diagram of an OPC compliant data server of FIGURE 3 constructed according to the principles of the present invention.

DETAILED DESCRIPTION

Referring initially to FIGURE 1, illustrated is a block diagram of a real-time process control system, generally designated 100, that forms one environment within which the present invention can operate. The real-time process control system 100 comprises a network 110 that interconnects a server 102, an operator interface 104 and a field unit 106. In the illustrated embodiment of the present invention, the real-time process control system 100 may comprise any number of servers 102, operator interfaces 104 and field units 106.

The network 110 comprises an industry standard network and industry standard network protocols. In the illustrated embodiment, the industry standard network is "10

base T," employing twisted pair cables. Other embodiments of the present invention use other networks comprising "10 base 2" employing coaxial cables, fiber optic cables or a combination of the two. Wireless communications may also be used for all or part of the network communications. The industry standard network protocols, in one embodiment of the present invention, are ETHERNET® and Transmission Control Protocol/Internet Protocol ("TCP/IP").

The server 102 comprises software programs that monitor, process information, and control the physical devices within the real-time process control system 100. The software programs comprise a requesting program "client," and a resource program "object" and other miscellaneous programs. The client program sends requests to object programs to perform specific functions. The object programs receive requests and perform the appropriate functions based upon the type of requests sent. The client programs and object programs communicate over the network 110 or internally within the server 102.

The operator interface 104 comprises a computer and a display. The operator interface 104 displays information concerning the current state of the system 100. The operator interface 104 also accepts operator input to perform functions such as controlling a physical device or requesting other information to be displayed on the operator interface's 104 display. The operator interface 104 may comprise both client programs and object programs. The operator interface 104 communicates to other programs over the network 110.

The field unit 106 comprises object programs that perform tasks related to the physical devices that make up the real-time process control system 100. In one embodiment of the present invention, the field unit's object programs collect status information, process data and control the physical devices. In other embodiments, the field unit 106 may perform more or less functions than described above. The field unit 106 responds to client's requests over the network 110.

Turning now to FIGURE 2, illustrated is a block diagram of a real-time process control software architecture, generally designated 200. The real-time process control software architecture 200 comprises an operator display software 202, a data processor software 204, an alarm processor software 206, a trend processor software 208, a scan processor software 220, a historical processor software 222, a report processor software

224, a field unit software 230, a database software 240 and the network 110 of FIGURE 1. In the illustrated embodiment of the present invention, the real-time process control software architecture 200 may comprise a plurality of the above software types.

The operator display software 202 displays the real-time process control system 100 information on a display or a plurality of displays. The operator display software 202 also processes the operator requests and communicates to other real-time process control software over the network 110.

The data processor software 204 processes the data collected and the data generated from the real-time process control system 100. The data processor software 204 stores and retrieves data to the database software 240 and communicates to other real-time process control software over the network 110.

The alarm processor software 206 performs alarm processing on the data collected. The alarm processor software 206 notifies the operator display software 202 and the report processor software 224 of any alarm conditions or non-alarm conditions that exist in the real-time process control system 100. The alarm processor software 206 also stores and retrieves information from the database software 240 over the network 110.

The trend processor software 208 performs trending functions for the real-time process control system 100. The trend processor software will collect operator selected data, generate the desired trend information and distribute the trend data to the operator display software 202 and the database software 240 over the network 110.

The scan processor software 220 collects data from a plurality of field units 230 and converts the data into the appropriate form usable by the real-time process control system 100. The scan processor software 220 distributes, over the network 110, the collected data to the other software processors so the software processors can perform their associated functions. The scan processor software 220 also stores and retrieves information from the database software 240.

The field unit 230 collects the specific data from the physical devices attached to the field unit 230. The physical devices are not shown since there are a multitude of physical devices that can be monitored by a real-time process control system. The field unit 230 sends the physical device data to the scan processor software 220. The field unit 230 also processes control requests.

The historical processor software 222 collects and processes historical information about the real-time process control system 100. The historical processor software 222 also performs archival functions and stores information to the database software 240.

The report processor software 224 generates the reports for the real-time process control system 100. The report processor software 224 sends the generated reports to the operator display software 202, the historical processor software 222, the database software 240 and to printing devices if attached to the system 100.

The database software 240 processes all request for retrieval and storage of information for the real-time process control system 100. In other embodiments of the present invention, the system 100 comprises a plurality of database software units contained on a plurality of computers.

Those skilled in the art should know that other embodiments of the present invention may include a plurality of processing software described above. Also, other embodiments of the present invention may include more or less processing software types and contain more or less functional capabilities than described above.

Turning now to FIGURE 3, illustrated is a block diagram of the overall process of creating an OPC compliant data server according to the principles of the present invention, generally designated 300. The user of the present invention starts with a template 310 and a device specific interface 320. The template 310 is a collection of source code for data server software routines such as the ones disclosed in the microfiche appendix submitted with this application and are incorporated herein by reference. The user then modifies the template 310 to meet the user's specific real-time process control requirements.

The device specific interface 320 is the user's software program that interfaces with the user's specific device or devices. The user combines the template 310 and the device specific interface 320 to produce a device specific noncompliant user application 330.

The device specific noncompliant user application 330 performs the user's required functions, but is not OPC compliant. In the illustrated embodiment of the present invention, the invention also comprises a server base dynamically linked library ("server base DLL") 335. The server base DLL 335 comprises software routines, when

implemented with the device specific noncompliant user application, allows the device specific noncompliant user application 330 to perform data server functions. The advantage of using the server base DLL 335, is that the user saves time by not programing the data server functions.

The present invention also comprises a set of software programs that implement the OPC/COM interface standards in the form of a generic OPC compliant interface object 340. Since the generic OPC compliant interface object 340 is also COM compliant, the generic OPC compliant interface object 340 is capable of self-registration.

In the illustrated embodiment of the present invention, there is aggregation between the device specific noncompliant user application 340 and the generic OPC compliant interface object 340. Aggregation is a method, under the COM standard, to implement object inheritance associated with object oriented distributed processing. In aggregation, an outer object passes the inner component's interface pointer directly to the client program instead of reimplementing all the functions of the interface and then forwarding the call to the inner component. In the present invention, the interface of the device specific noncompliant user application 330 aggregates to the generic OPC compliant interface object 340.

Background information concerning COM aggregation is discussed in Inside COM by Dale Rogerson, Microsoft Press 1997 and in Essential COM by Don Box, Addison-Wesley 1998. The foregoing publications are incorporated herein by reference.

Next, the user links the base server DLL 335 and the generic OPC compliant interface object 340 with the device specific noncompliant user application 330 to produce the final OPC compliant data server 350. The OPC compliant data server 350 is the data server for a specific device. The above process is repeated for each device in the real-time process control system 100 requiring an OPC compliant data server.

In the illustrated embodiment of the present invention, the generic OPC compliant interface object 340 is a set of dynamically linkable libraries ("DLLs"). When the user links the DLLs, the DLLs are not incorporated into the OPC compliant data server 350 executable software. The OPC compliant data server 350 executable software contains operating system calls that will load the DLLs when needed.

The advantage of using DLLs for the OPC/COM compliant interface software and the server base software is that when new updates to the OPC/COM compliant interface software or the server base software are distributed, the user does not have to recompile all the software they have written. The user merely loads the new DLLs on the machine where the OPC compliant data server 350 is located and reinitialize the software program. Thus, the present invention saves the user a tremendous amount of time and effort by not requiring the user to recompile the user's software programs each time a correction or enhancement is made to the OPC/COM compliant interface software.

Those skilled in the art should note that the present invention is not restricted to the use of a generic OPC compliant interface object in the form of DLLs. The use of a generic OPC compliant interface object may be implemented on different types of operating systems that may or may not use DLLs. Also, those skilled in the art should realize that the procedure described above is an overview of the use of the present invention and the present invention is not limited to the procedure described above.

Turning now to FIGURE 4, illustrated is a block diagram of an OPC compliant data server of FIGURE 3 constructed according to the principles of the present invention. The OPC compliant data server 350 comprises one or more instances of a device specific server 410, a cache memory 420 and a class factory 430. Each instance of the device specific server 410 comprises the device specific noncompliant user application 330 and the generic OPC compliant interface object 340. See Figure 3 for a description of the use and advantages of the generic OPC compliant interface object 340. Each instance of the device specific server 410 processes client requests and communicates with the appropriate physical device.

The cache memory 420 comprises a cached storage area used by the OPC compliant data server 350. In one embodiment of the present invention, a client program requests information from the OPC compliant data server 350. The generic OPC compliant interface object 340 then retrieves the appropriate data from the device through the device specific noncompliant user application 330, returns the data to the client and stores the data in the cache memory 420. Upon subsequent requests from clients, the generic OPC compliant interface object retrieves the data from the cache memory 420. The cache memory 420 increases throughput and performance of the

OPC compliant data server 350.

In another embodiment of the present invention, for each client information request, the generic OPC compliant interface object 340 retrieves the appropriate data from the cache memory 420 and returns the data to the client. Upon a predetermined interval, the generic OPC compliant interface object 340 retrieves the appropriate data from the device through the device specific noncompliant user application 330 and updates the data in the cache memory 420.

The class factory 430, which is part of the server base DLL 335, creates and manages the device specific server 410 instances and performs class registration under the COM standard. The class factory also manages the other resources associated with each instance of the device specific server 410.

Those skilled in the art should note that the OPC compliant data server may contain more or less features than described above. Also, the OPC compliant data server is not limited to Class registration under the COM standard and may be implemented on different operating systems.

Although the present invention has been described in detail, those skilled in the art should understand that they can make various changes, substitutions and alterations herein without departing from the spirit and scope of the invention in its broadest form.

CLAIMS

1. A system for constructing an Object Linking and Embedding (OLE) for Process Control (OPC) compliant data server from a device-specific noncompliant user application, comprising:

a generic OPC compliant interface object; and

a template associated with said generic OPC compliant interface object, said template being modifiable and combinable with a device-specific interface portion to yield said device-specific noncompliant user application, said device-specific noncompliant user application dynamically linkable with and aggregating to said generic OPC compliant interface object to yield said OPC compliant data server.

2. The system as recited in Claim 1 further comprising a class factory that creates and manages multiple related instances of objects.

3. The system as recited in Claim 1 further comprising a class factory that creates instances of objects and manages global resources with respect to said objects.

4. The system as recited in Claim 1 further comprising a cache memory associated with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from said cache memory in response to requests from multiple OPC clients.

5. The system as recited in Claim 1 further comprising a cache memory associated with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from a device through said device-specific noncompliant user application and storing said items in said cache memory.

6. The system as recited in Claim 2 wherein said class factory is capable of self-registration.

7. The system as recited in Claim 1 wherein said generic OPC compliant interface object is Component Object Module (COM) compliant.

8. The system as recited in Claim 1 further comprising a COM interface implemented by an aggregating COM object and called by an aggregated COM object.

9. A method of constructing an Object Linking and Embedding (OLE) for Process Control (OPC) compliant data server from a device-specific noncompliant user application, comprising:

providing a generic OPC compliant interface object;

modifying and combining a template with a device-specific interface portion to yield said device-specific noncompliant user application; and

dynamically linking said device-specific noncompliant user application with said generic OPC compliant interface object to yield said OPC compliant data server, said device-specific noncompliant user application aggregating to said generic OPC compliant interface object.

10. The method as recited in Claim 9 further comprising providing a class factory that creates and manages multiple related instances of objects.

11. The method as recited in Claim 9 further comprising providing a class factory that creates instances of objects and manages global resources with respect to said objects.

12. The method as recited in Claim 9 further comprising associating a cache memory with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from said cache memory in response to requests from multiple OPC clients.

13. The method as recited in Claim 9 further comprising associating a cache memory with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from a device through said device-specific noncompliant user application and storing said items in said cache memory.

14. The method as recited in Claim 10 further comprising self-registration of objects with said class factory.

15. The method as recited in Claim 9 wherein said generic OPC compliant interface object is Component Object Module (COM) compliant.

16. Method as recited in Claim 9 further comprising implementing a COM interface by an aggregating COM object and calling said COM interface by an aggregated COM. object.

17. A real-time process control system, comprising:

at least one device capable of generating data;

a system for constructing an Object Linking and Embedding (OLE) for Process Control (OPC) compliant data server for said data from a device-specific noncompliant user application, including:

a generic OPC compliant interface object, and

a template associated with said generic OPC compliant interface object, said template being modifiable and combinable with a device-specific interface portion to yield said device-specific noncompliant user application, said device-specific noncompliant user application dynamically linkable with and aggregating to said generic OPC compliant interface object to yield said OPC compliant data server; and

a client, couplable to said OPC compliant data server, that makes a request of said OPC compliant data server for said data.

18. The system as recited in Claim 17 wherein said system further includes a class factory that creates and manages multiple related instances of objects.

19. The system as recited in Claim 17 wherein said system further includes a class factory that creates instances of objects and manages global resources with respect to said objects.

20. The system as recited in Claim 17 wherein said system further includes a cache memory associated with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from said cache memory in response to requests from multiple OPC clients.

21. The system as recited in Claim 17 wherein said system further includes a cache memory associated with said generic OPC compliant interface object, said generic OPC compliant interface object retrieving items from a device through said device-specific noncompliant user application and storing said items in said cache memory.

22. The system as recited in Claim 18 wherein said class factory is capable of self-registration.

23. The system as recited in Claim 17 wherein said generic OPC compliant interface object is Component Object Module (COM) compliant.

24. The system as recited in Claim 17 further comprising a COM interface implemented by an aggregating COM object and called by an aggregated COM object.

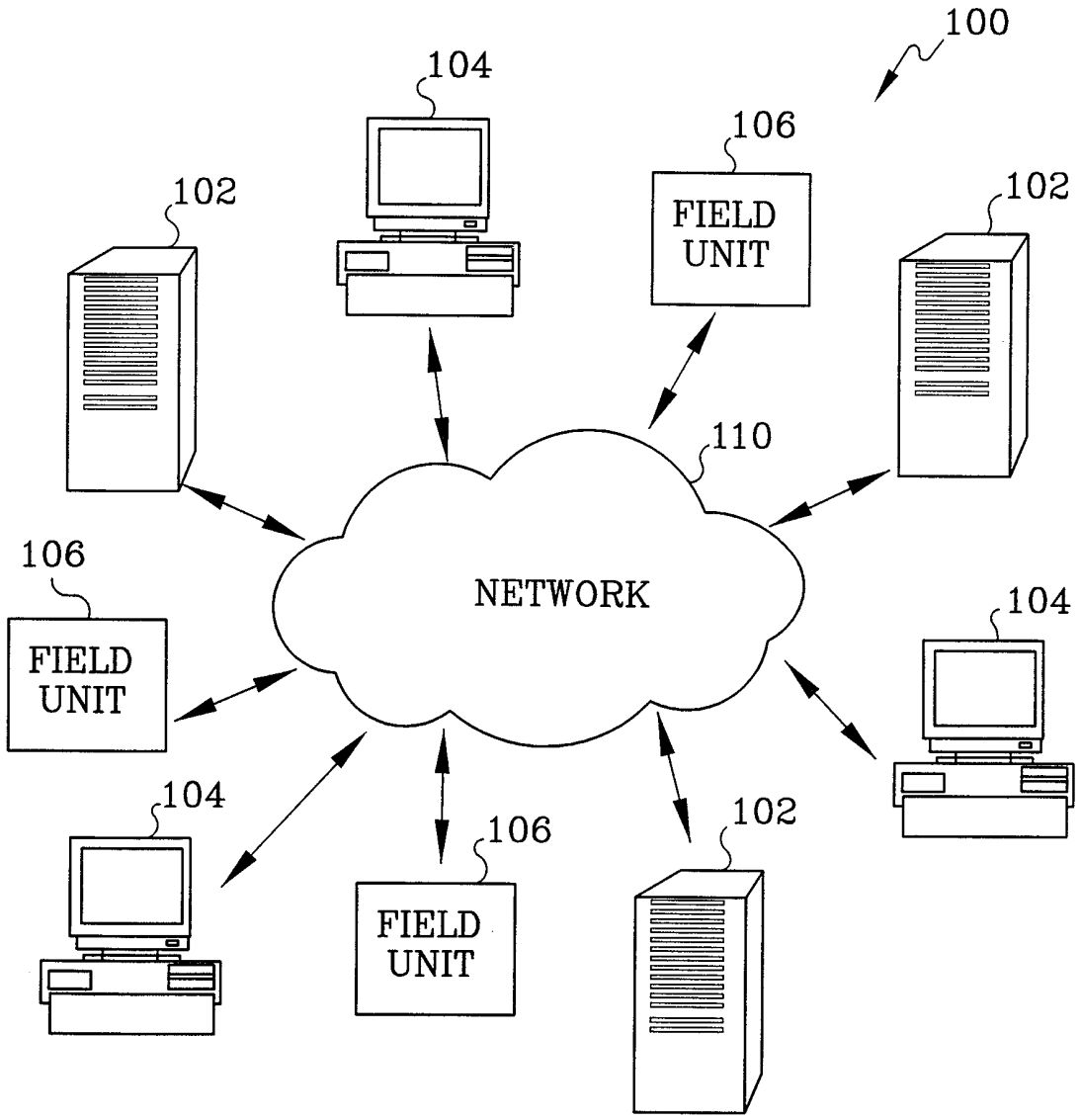

100
104
106
FIELD UNIT
102
102
110
NETWORK
106
FIELD UNIT
104
104
106
FIELD UNIT
102

Fig. 1

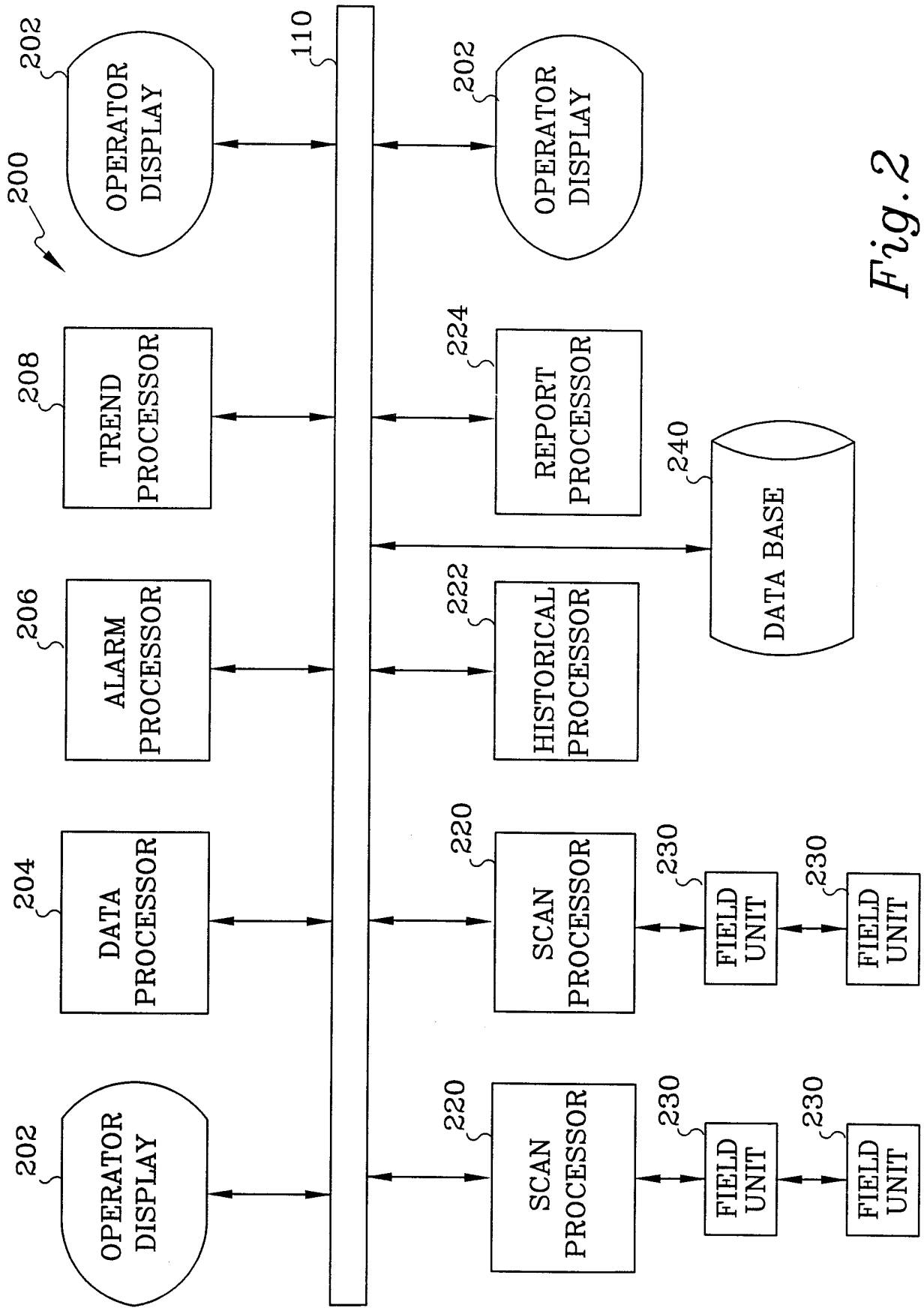

200
202
OPERATOR DISPLAY
110
202
OPERATOR DISPLAY
208
TREND PROCESSOR
224
REPORT PROCESSOR
240
DATA BASE
206
ALARM PROCESSOR
222
HISTORICAL PROCESSOR
204
DATA PROCESSOR
220
SCAN PROCESSOR
230
FIELD UNIT
230
FIELD UNIT
202
OPERATOR DISPLAY
220
SCAN PROCESSOR
230
FIELD UNIT
230
FIELD UNIT

Fig. 2

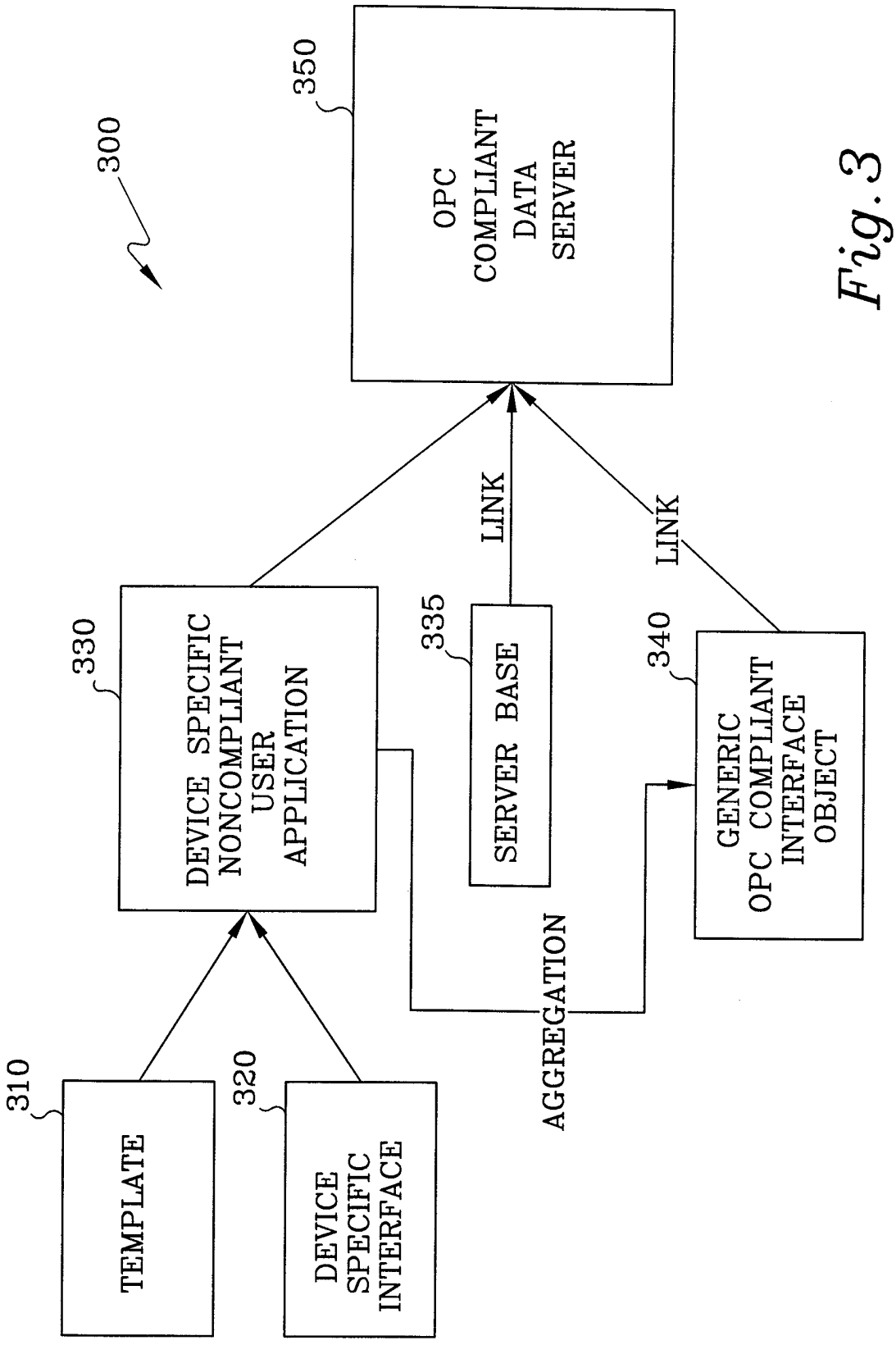

300
350
OPC COMPLIANT DATA SERVER
LINK
LINK
330
DEVICE SPECIFIC NONCOMPLIANT USER APPLICATION
335
SERVER BASE
340
GENERIC OPC COMPLIANT INTERFACE OBJECT
AGGREGATION
310
TEMPLATE
320
DEVICE SPECIFIC INTERFACE

Fig. 3

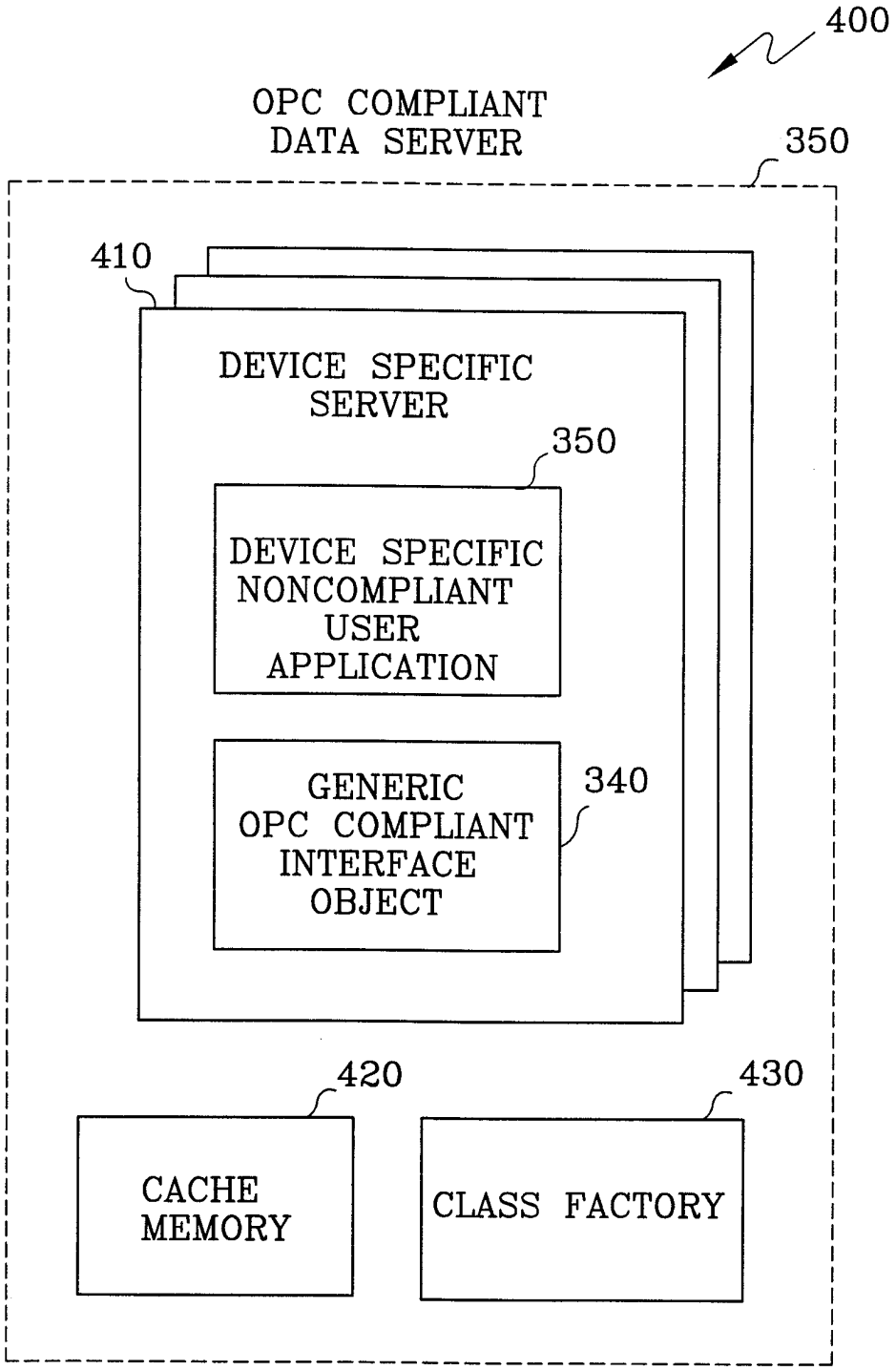

400
OPC COMPLIANT
DATA SERVER
350
410
DEVICE SPECIFIC
SERVER
350
DEVICE SPECIFIC
NONCOMPLIANT
USER
APPLICATION
GENERIC
OPC COMPLIANT
INTERFACE
OBJECT
340
420
430
CACHE
MEMORY
CLASS FACTORY

Fig. 4