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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 June 2004 (24.06.2004)

PCT

(10) International Publication Number
WO 2004/053601 A2

(51) International Patent Classification[7]: G05B 15/00

(21) International Application Number:
PCT/US2003/039041

(22) International Filing Date: 8 December 2003 (08.12.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/315,857 10 December 2002 (10.12.2002) US

(71) Applicant: **FISHER-ROSEMOUNT SYSTEMS, INC.** [US/US]; 8301 Cameron Road, Austin, TX 78754-3895 (US).

(72) Inventors: **DOBROWSKI, Patrick, M.**; 408 Oakland Lane, Brunsville, MN 55337 (US). **HOKENESS, Scott, N.**; 17335 Jade Terrace, Lakeville, MN 55044 (US). **MIDDENDORF, Fred, G.**; 4248 Trenton Road, Eagan, MN 55123 (US).

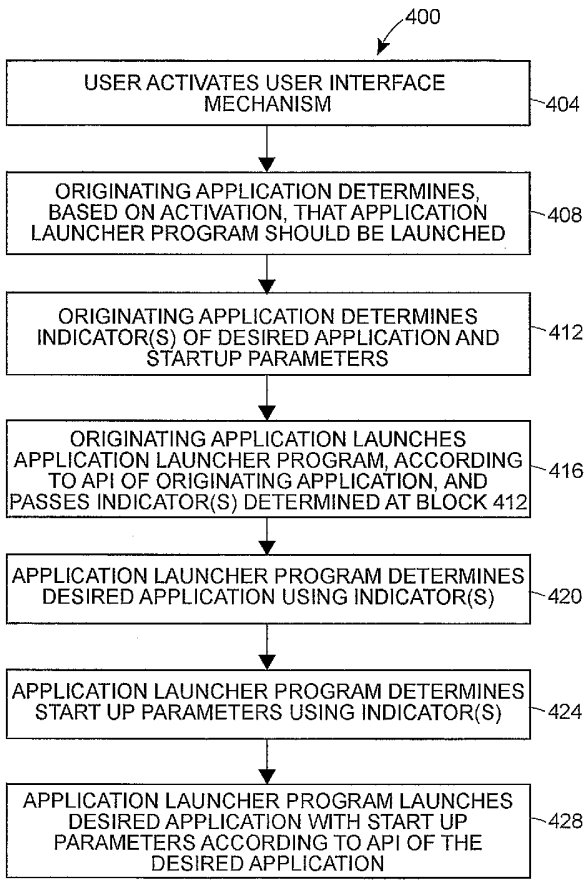
(74) Agents: **HEPPERMANN, Roger, A.** et al.; Marshall, Gerstein & Borun LLP, 233 S. Wacker Drive, 6300 Sears Tower, Chicago, IL 60606-6357 (US).

(81) Designated States *(national)*: 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, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE,

[Continued on next page]

(54) Title: METHOD FOR LAUNCHING APPLICATIONS



(57) Abstract: An originating application in a process plant is able to launch multiple desired applications, with one or more of the desired applications having a different application programming interface. In one example, the originating application is able to launch multiple desired applications via an application launcher program. The originating application provides the application launcher program with an indication of the desired application to be launched, and the application launcher program then launches the indicated desired application.

WO 2004/053601 A2

ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that 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.

METHOD FOR LAUNCHING APPLICATIONS

FIELD OF THE DISCLOSURE

[01] The present disclosure generally relates to process plants, and, more particularly, to launching applications from an originating application in a process plant.

BACKGROUND

[02] Process plants, like those used in chemical, petroleum, manufacturing or other processes, typically include one or more process controllers communicatively coupled to one or more field devices via analog, digital or combined analog/digital buses. The field devices, which may be, for example, valves, valve positioners, switches and transmitters (e.g., temperature, pressure and flow rate sensors), are located within the process environment and perform process functions such as opening or closing valves, measuring process parameters, etc. Smart field devices, such as the field devices conforming to a well-known fieldbus protocol (e.g. the Foundation Fieldbus protocol, the HART protocol, etc.) may also perform control calculations, alarming functions, and other control functions commonly implemented within the controller. The process controllers, which are also typically located within the plant environment, receive signals indicative of process measurements made by the field devices and/or other information pertaining to the field devices and execute a controller application that runs, for example, different control modules which make process control decisions, generate control signals based on the received information and coordinate with the control modules or blocks being performed in the field devices, such as HART and Foundation Fieldbus field devices. The control modules in the controller send the control signals over the communication lines to the field devices to thereby control the operation of the process.

[03] Information from the field devices and the controller is usually made available over a data highway to one or more other hardware devices, such as operator

workstations, personal computers, data historians, report generators, centralized databases, etc., typically placed in control rooms or other locations away from the harsher plant environment. These hardware devices run applications that may, for example, enable an operator to perform functions with respect to the process, such as changing settings of the process control routine, modifying the operation of the control modules within the controller or the field devices, viewing the current state of the process, simulating the operation of the process for the purpose of training personnel or testing the process control software, keeping and updating a configuration database, etc.

[04] As an example, control systems, such as the DeltaV™ control system sold by Emerson Process Management, may include multiple applications stored within and executed by different hardware devices located at diverse places within a process plant. For instance, a control system may include a configuration application that resides in one or more workstations. The configuration application may enable users to create or change process control modules and download these process control modules via a data highway to dedicated distributed controllers. The configuration application may also allow a designer to create or change user interfaces which are used by a viewing application to display data to a user and to enable a user to change settings, such as set points, within the process control routine.

[05] Similarly, the control system may include a configuration database application that may be executed, for example, by the same workstation(s) which execute the configuration application, or by a different hardware device coupled to the data highway. The configuration database application may store the current process control routine configuration and data associated therewith in a configuration database. Also, the control system may include a data historian application that may be executed by a data historian device. The data historian application controls the data historian device to collect and store some or all of the data provided across the data highway.

[06] Additionally, a control system may include controller applications that may be stored and executed on a dedicated controller and, in some cases, field devices. A controller application may run control modules, which have been assigned and downloaded to the controller or field device, to implement actual process control.

[07] Further, a control system may include viewing applications that may be executed on one or more operator workstations. A viewing application may receive data, for example, from a controller application via a data highway and display this data to process plant designers, operators, or users using user interfaces. The user interfaces may provide any of a number of different views, such as an operator's view, an engineer's view, a technician's view, etc.

[08] As another example, management systems, such as the AMS predictive maintenance software sold by Emerson Process Management, may interact with one or more smart field devices to read the device, block, parameter, variable, or configuration information associated with those devices. Typically, a management system may reside on one or more operator workstations having appropriate communication ports which allow it to interconnect to, communicate with, and reconfigure a smart device. Management systems may be on-line, that is, have a hard-wired or any other permanent connection with a smart device. Management systems may also be portable and be capable of being periodically connected to a smart device to reconfigure or troubleshoot problems with that smart device.

[09] Management systems typically perform a wide variety of functions with respect to smart devices within a system. For example, management systems may be used to provide users with information (e.g., values of variables or parameters) pertaining to the state of a process and to each of the smart field devices associated with or connected to the process. Management systems may also be used to enable a user to monitor a process and control the process by reconfiguring smart devices within the process as necessary.

[10] A management system may include one or more core applications as well as add-on applications provided, for example, by individual smart device manufacturers to implement changes on, and read data from, a particular smart device. A core application may, for example, include a graphical user interface that provides hierarchical views of devices in a plant or process. A core application may also, for example, provide a base functionality for interfacing with certain devices. For instance, a core application may permit a user to configure and/or read data from HART devices at a generic level. An add-on application may, for example, permit a user to configure and/or read data from devices that communicate via protocol other

than that (those) supported by a core application. Also, an add-on application may, for example, permit a user to configure and/or read data from devices at level beyond the generic level supported by a core application. Further, management systems may utilize other applications such as word processing programs, spreadsheet programs, internet browsing programs, etc. For example, a management system may utilize an internet browsing program to display on-line maintenance documents made available by a device manufacturer.

[11] Applications in a process plant often launch other applications. For example, a management system may launch a first application provided by a first smart device manufacturer to configure a first smart device. Also, the management system may launch a second application provided by a second smart device manufacturer to configure a second smart device. Similarly, a management system may launch a web browser to display a web page that provides information about a particular device.

[12] Various applications to be launched often do not share a common or consistent application program interface (API). Therefore, an originating application that launches multiple other applications, often includes multiple custom launching mechanisms. Developing such multiple launching mechanisms can be cumbersome and time consuming. Additionally, if it is desired to provide a new application that can be launched from an existing originating application, a new custom launching mechanism is typically developed, and the originating application modified to include the new custom launching mechanism.

SUMMARY OF THE DISCLOSURE

[13] The examples described herein are generally related to launching desired applications via an originating application in a process plant. It may be desired for an originating application to be able to launch multiple desired applications, with one or more of the desired applications having a different application programming interface. In one example, the originating application is able to launch multiple desired applications via an application launcher program, subroutine, object, etc. The originating application provides the application launcher

program, subroutine, object, etc. with an indication of the desired application to be launched, and the application launcher program, subroutine, object, etc. then launches the indicated desired application.

[14] A convenient user interface may be used to configure the process plant application to launch multiple desired applications via the application launcher program. In one example, a user may be prompted to indicate at least one desired application. For instance, the user may be prompted to specify a name of an executable file of a desired application, and may also be prompted to specify a location (e.g., a path) of the executable file. Additionally, the user may be prompted to indicate a component in a process plant, with which the at least one desired application is to be associated. In the case of a management system application, the user may be prompted to specify a device type with which the desired application is to be associated. Then, the at least one desired application may be associated with the at least one component. For instance, an indication of the desired application may be stored in a file associated with a specified component of the process plant.

BRIEF DESCRIPTION OF THE DRAWINGS

[15] The features and advantages of the examples described herein will be best appreciated upon reference to the following detailed description and the accompanying drawings, in which:

[16] Fig. 1 is a block diagram of an example process plant that may utilize techniques described herein for launching a desired application;

[17] Fig. 2 is a block diagram of an example workstation that may be included in the example process plant of Fig. 1;

[18] Fig. 3 is a block diagram of an example process plant application that may utilize an application launcher program;

[19] Fig. 4 is a flow diagram of an example method for launching a desired application via an application launcher program;

[20] Fig. 5 is a block diagram illustrating an example technique for launching a desired application via an application launcher program;

[21] Fig. 6 is a flow diagram of an example method for configuring a process plant application for launching a desired application via an application launcher program;

[22] Fig. 7 is an example user interface for configuring a process plant application for launching a desired application via an application launcher program;

[23] Fig. 8 is an example user interface for configuring a process plant application for launching a desired application via an application launcher program;

[24] Fig. 9 is an example user interface for configuring a process plant application for launching a desired application via an application launcher program;

[25] Fig. 10 is a flow diagram of an example method for configuring a management system application for launching a desired application via an application launcher program;

[26] Fig. 11 is an example user interface for a management system application;

[27] Fig. 12 is a flow diagram of an example method for launching a desired application via an application launcher program; and

[28] Fig. 13 is a block diagram illustrating an example technique for launching a desired application via an application launcher program.

DETAILED DESCRIPTION

[29] For ease of illustration, some of the following examples are described in the context of a management software application in a process plant. It will be appreciated by one of ordinary skill in the art, however, that the techniques described herein can be used in other aspects of a process plant such as a control system application, a device or equipment management application, a simulation application,

documentation system, work order management system, enterprise resource planning system, procurement system, etc.

[30] **System Overview**

[31] Fig. 1 is a block diagram of an example process plant 10. The process plant 10 includes one or more nodes 12, 14, 16, 18 and 20. In the example process plant 10 of Fig. 1, each of the nodes 12, 14 and 16 includes a process controller 12a, 14a, 16a connected to one or more field devices 22 and 23 via input/output (I/O) devices 24 which may be, for example, Foundation Fieldbus interfaces, HART interfaces, etc. The controllers 12a, 14a and 16a are also coupled to one or more host or operator workstations 18a and 20a in the nodes 18 and 20 via a data highway 30 which may be, for example, an Ethernet link. While the controller nodes 12, 14, 16 and the input/output and field devices 22, 23 and 24 associated therewith are typically located down within and distributed throughout the sometimes harsh plant environment, the operator workstation nodes 18 and 20 are usually located in control rooms or other less harsh environments easily assessable by controller personnel.

[32] Generally speaking, the workstations 18a and 20a of the nodes 18 and 20 may be used to store and execute applications used to configure and monitor the process plant 10, and to manage devices 22, 23 in the process plant 10. Further, a database 32 may be connected to the data highway 30 and operate as a data historian and/or a configuration database that stores the current configuration of the process plant 10 as downloaded to and/or stored within the nodes 12, 14, 16, 18 and 20.

[33] Each of the controllers 12a, 14a and 16a, which may be by way of example, the DeltaV controller sold by Emerson Process Management, may store and execute a controller application that implements a control strategy using a number of different, independently executed, control modules or blocks. The control modules may each be made up of what are commonly referred to as function blocks wherein each function block is a part or a subroutine of an overall control routine and operates in conjunction with other function blocks (via communications called links) to implement process control loops within the process plant 10. As is well known, function blocks typically perform one of an input function (such as that associated with a transmitter, a sensor or other process parameter measurement device), a control

function (such as that associated with a control routine that performs PID, fuzzy logic, etc. control), or an output function which controls the operation of some device (such as a valve, to perform some physical function within the process plant 10). Of course hybrid and other types of function blocks exist and may be utilized. While a fieldbus protocol and the DeltaV system protocol may use control modules and function blocks designed and implemented in an object oriented programming protocol, the control modules could be designed using any desired control programming scheme including, for example, sequential function block, ladder logic, etc. and are not limited to being designed using function block or any other particular programming technique. As is typical, the configuration of the control modules as stored within the process control nodes 12, 14 and 16 is stored in the configuration database 32 which is accessible to applications executed by the workstations 18a and 20a.

[34] In the system illustrated in Fig. 1, the field devices 22 and 23 coupled to the controllers 12a, 14a and 16a may be standard 4-20 ma devices, or may be smart field devices, such as HART, Profibus, or Foundation Fieldbus field devices, which include a processor and a memory. Some of these devices, such as Foundation Fieldbus field devices (labeled with reference number 23 in Fig. 1), may store and execute modules, or sub modules, such as function blocks, associated with the control strategy implemented in the controllers 12a, 14a and 16a. Of course, the field devices 22, 23 may be any types of devices, such as sensors, valves, transmitters, positioners, etc. and the I/O devices 24 may be any types of I/O devices conforming to any desired communication or controller protocol such as HART, Foundation Fieldbus, Profibus, etc.

[35] A management application may be stored on and executed by one or more of workstations 18a and 20a. Fig. 2 is a block diagram of example workstation 18a (workstation 20a may comprise a same or similar device). The workstation 18a may include at least one processor 50, a volatile memory 54, and a non-volatile memory 58. The volatile memory may include, for example, a random access memory (RAM). The non-volatile memory may include, for example, one or more of a hard disk, a read-only memory (ROM), a compact disk ROM (CD-ROM), a digital versatile disk (DVD), a FLASH memory, etc. The workstation 18a may also include an input/output (I/O) device 62. The processor 50, volatile memory 54, non-volatile

memory 58, and I/O device 62 may be interconnected via an address/data bus 66. The workstation 18a may also include at least one display 70 and at least one input device 74. The input device may include, for example, one or more of a keyboard, a keypad, a mouse, etc.

[36] The display 70 and the input device 74 are coupled with the I/O device 62. Additionally, the workstation 18a is coupled with the data highway via the I/O device 62. Although I/O device 62 is illustrated in Fig. 2 as one device, it may comprise several devices.

[37] A management application may be stored, for example, in whole or in part, in the non-volatile memory 58 and executed, in whole or in part, by the processor 50.

[38] **Application Launcher Overview**

[39] Fig. 3 is a block diagram of an example process plant application 100. For ease of explanation, Fig. 3 will be described in the context of a management system application. It is to be understood, however, that the process plant application 100 may be any of various types of application employed in a process plant (e.g., a control system application, a device or equipment management application, a simulation application, etc.). The process plant application 100 may include one or more core applications 104, and may also include one or more add-on applications 108 and/or one or more custom applications 112. In a management system application, a core application 104 may, for example, provide a base functionality for communicating with and configuring certain types of devices (e.g., HART devices, Fieldbus devices, etc.) and/or provide a graphical user interface for displaying devices in a process plant.

[40] In a management system, an add-on application 108 may be, for example, an application, provided by a manufacturer of a device, that permits communication and configuration resources for a device for which the core application 104 does not provide such resources. Similarly, an add-on application 108 may be, for example, an application, provided by a manufacturer of a device, that permits communication and configuration resources for a device beyond those provided by the core application 104. In a management system, a custom application

may be, for example, an application developed by an end user that provides a level of communication and configuration resources not provided by a core application 104 or an add-on application 108.

[41] The process plant application 100 may also utilize one or more general purpose applications 116, such as a word processing application, a spreadsheet application, a web browsing application, etc. The process plant application 100 may further include an application launcher program 120.

[42] An application, such as core application 104, can launch applications, such as add-on applications 108, custom applications 112, and general purpose applications 116, via the application launcher program 120. In the example illustrated in Fig. 3, core application 104 uses application launcher program 120 to launch other applications such as add-on applications 108, custom applications 112, and general purpose applications 116. The core application 104, however, need not use application launcher 120 to launch every application. Additionally, in some examples, other applications besides core application 104 may utilize the application launcher program 120 or a separate application launcher program (not illustrated) similar to application launcher program 120. For example, an add-on application 108 may launch a general purpose application 116 via a separate application launcher program (not illustrated).

[43] Each of the add-on applications 108, the custom applications 112, and the general purpose applications 116, potentially, may have a unique API. Thus, by utilizing the application launcher program 120, the core application 104 may launch multiple applications, having multiple APIs, via potentially only a single API.

[44] Fig. 4 is a flow diagram illustrating an example method 150 of launching an application via an application launcher program. For ease of explanation, the flow of Fig. 4 will be described with reference to Fig. 5, which is a block diagram illustrating an example operation of an originating application 180 utilizing the application launcher program 120 to launch a desired application 184. Additional examples of methods for utilizing an application launcher program will be described below.

[45] At block 154, an originating application 180 launches the application launcher program 120. The originating application 180 may be, for example, a core application of a process plant application. When launching the application launcher program 120, the originating application 180 may provide to the application launcher program 120 an indication of the particular desired application 184 to be launched. Additionally, the originating program 180 may provide to the application launcher program 120 an indication of startup parameters which may be used to launch the desired application 184. It will be appreciated by one of ordinary skill in the art that the indication of the desired application 184 and the indication of startup parameters may be a single indicator or multiple indicators.

[46] The launching of the application launcher program 120 and the providing of the indicator or indicators of the desired application 184 and the startup parameters may be performed according to an API of the application launching program 120. It will be appreciated by those of ordinary skill in the art that the API of the application launching program 120 may be designed to match a portion of the API of the originating application 180 that corresponds to program launching.

[47] At block 158, the application launching program 120 may utilize the indicator or indicators of the desired application 184 and the startup parameters to retrieve information 188 related to launching the desired application 184 according to the API of the desired application 184. The information 188 may include, for example, a file name of the desired application 184, a location of the desired application 184 (e.g., a path name), startup parameters to be provided to the desired application 184 in an order and syntax according to the API of the desired application 184 (e.g., a command line), etc. The information 188 may be stored, for example, in a file, database, etc., such that the application launching program 120 can use the indicator or indicators of the desired application 184 and the startup parameters to retrieve the information 188. The indicator(s) may include, for example, a file name in which the information 188 is stored, a location within a file, a database query for retrieving the information 188 from a database, a URL, an XML document, a HTTP GET, etc.

[48] At block 162, the application launcher program 120 utilizes the information 188 to launch the desired application according to an API of the desired

application 184. If startup parameters are provided, the desired application 184 may be considered to be launched "in-context." For example, if the desired application 184 is a web browser, the web browser may be launched displaying a particular web page.

[49] **Configuring the Process Plant Application**

[50] Fig. 6 is a flow diagram illustrating an example method 200 for configuring a process plant application in which an originating application can launch a desired application via an application launcher program. At block 204, a user may be prompted to provide a descriptive name for a desired application. The descriptive name may be used, for example, in providing a graphical user interface mechanism via which a user can initiate a launch of the desired application.

[51] At block 208, the user may be prompted to provide a file name and a location of an executable file of the desired application. At block 212, the user may be prompted to provide startup parameters for the desired application. The types of startup parameters that may be provided may be determined, in part, by the API of the desired application. Additionally, the user may specify, for example, an ordering, a syntax, etc., of the startup parameters that correspond to the desired application's API.

[52] At block 216, the user may be prompted to specify one or more components in a process plant with which the desired application should be associated. For instance, in a control system application, the user may specify one or more of a controller, a module, a device, etc., used in a process. In a management system application, the user may specify, for example, a specific device, a device type, etc.

[53] At block 220, the desired application and the startup parameters may be associated with the one or more components (which the user was prompted to specify at block 216). For instance, a text file may be created that includes the file name and path of the executable file and the startup parameters to be used in launching the desired application. The text file may be given a unique name that can be associated with the component(s) (which the user was prompted to specify at block 216).

[54] At block 224, the process plant application may be configured to reflect the association between the desired application (which the user was prompted to specify at block 208) and the component(s) (which the user was prompted to specify at block 216). For instance, an originating application of the process plant application may be configured to provide a user interface mechanism for initiating a launch of the desired application. The user interface mechanism may be associated with the component(s). As a specific example, a menu associated with the component(s) may be configured to include an item having the descriptive name which the user was prompted to specify at block 204. Continuing with the example described with reference to block 220, the menu item may be associated with the text file that includes the file name and path of the executable file and the startup parameters to be used in launching the desired application.

[55] **Configuring a Management System Application**

[56] Figs. 7-9 are an example user interface that may be utilized to implement blocks 204, 208, 212, and 216 of Fig. 6 for a management system application. In particular, Fig. 7 is an example user interface 250 that may be used for prompting a user to provide a descriptive name (block 204 of Fig. 6), a filename of an executable file for the desired application (block 208 of Fig. 6), and a location of the executable file (block 212 of Fig. 6). For instance, a user may type in a descriptive name for the desired application in box 254. In block 258, a user may type in a path and executable file name. Additionally, a user may "browse" for the path and executable file name in a manner well known to those of ordinary skill in the art by selecting button 262.

[57] Fig. 8 is an example user interface 250 that may be used for prompting a user to provide startup parameters for the desired application (block 212 of Fig. 6). The user interface includes boxes 272a—272m and 274a—274m for specifying parameters. Parameters can be passed to the desired application in order (e.g., "Argument 1", then "Argument 2", then "Argument 3", etc.). In this way, the user may select the order in which to pass the parameter to correspond, for example, with an API of the desired application. A user may type in parameters in boxes 272a—272m. Additionally, using boxes 274a—274m, a user may pass "Arguments" to the desired application. An argument may correspond, for example, to a type of

information corresponding to a device. In an example in which the management system application includes the AMS predictive maintenance software sold by Emerson Process Management, arguments may include AMS Username (the AMS login name of a user), OPC Block Moniker (a device's Object Linking and Embedding (OLE) for Process Control (OPC) format block moniker), Plant Server Name (the name of the server for a connected device), Device Connection State (indicates whether a device is "live", i.e., "connected" or "not connected"), AMS Block Tag (a device icon label for a device that may be displayed to the user in, for example, the AMS device connection view or the AMS plant server view), Device Block Key (a number used for database device references, commonly required for structured query language (SQL) database accesses), Database Name (the domain name server (DNS) name of the database for a device), Manufacturer ID (a number that indicates a device's manufacturer), Device Type (a number that indicates a device type or model), HART Revision (a number that indicates a HART protocol revision supported by a device), Device ID (a number that indicates a specific device for a given Manufacturer ID and Device Type), etc. The user interface 270 may include buttons 276a—276m to permit a user to select an argument from, for example, a pull-down menu such as menu 278. By specifying one or more arguments via boxes 274a—274m, a user may specify a type of information to be passed to the desired application, rather than specifying specific information.

[58] Using user interface 270, a person may specify an appropriate syntax for passing information to the desired application. For instance, using boxes 280b—280m, 272a—272m, and 282a—282m, a user may add spaces, delimiters, and/or quotation marks and create a "command line" that includes the parameters in the appropriate syntax. Box 284 may provide the user with a view of the command line as it is being created so, for example, the user can verify that the correct syntax is being used.

[59] Fig. 9 is an example user interface 290 that may be used for prompting a user to specify one or more devices with which the desired application should be associated (block 216 of Fig. 6). The user interface 290 may include a list 292 of possible devices and/or device types. A user may select devices and/or device types to be associated with the desired application by selecting buttons 296a and 298a. For

example, by selecting button 296a, the user may select all of the devices and/or device types in list 292. Additionally, the user may highlight one or more individual devices and/or device types in list 292, and then select those one or more devices and/or device types by selecting button 298a.

[60] The user interface 290 may also include a list 294 of devices and/or device types that have been selected to be associated with the desired application. The user may use button 296b to "deselect" all of the devices and/or device types in list 294. Additionally, the user may highlight one or more individual devices and/or device types in list 294, and the select button 298b to "deselect" those devices and/or device types.

[61] Because the user interfaces 250, 270, and 290 may permit a user to associate a desired application with a device type, the user, in effect, may be able to associate the desired application with, potentially, multiple devices. Thus, the user need not associate the desired application with each device, individually. In other examples, the user may be permitted to associate desired applications with individual devices.

[62] Continuing with the example described with reference to Figs. 7-9, associating the desired application and startup parameters with the one or more devices (block 220 of Fig. 6) may include storing an indication of the association in a file, such as a configuration file. If the management system application is implemented with a WINDOWS™ operating system available from Microsoft Corporation (e.g., WINDOWS XP™, WINDOWS 2000™, WINDOWS NT™, etc.), indication(s) of the association may be stored, for example, in an INI file and/or a Registry file. For instance, the information obtained via the user interface described with reference to Figs. 7-9 may be stored in an INI file.

[63] Fig. 10 is a flow diagram of an example method 320 for configuring the management system application (block 224 of Fig. 6). At block 324, information that will enable the application launcher program to launch the desired program is formatted according to the standard API of the originating program. Alternatively, the information may be formatted according to an API that is not standard for the

originating program. For example, a standard API of the application program launcher could be employed.

[64] In one specific example, the desired application's executable file and path may be stored in a file. Additionally, parameters assembled into a command line via, for example, a user interface such as user interface 270 (Fig. 8) may also be stored in the file.

[65] At block 328, an originating application of the management system application is configured to include one or more user interface mechanisms for initiating a launch of the desired application. For instance, if the user specified one or more device types via user interface 290 (Fig. 9), and the process or plant includes multiple devices of that type or types, multiple user interface mechanisms corresponding to the multiple devices may be included.

[66] Fig. 11 is a display from an example originating application. In this example, the originating application provides a hierarchical view of devices in a process and/or plant. The display 350 includes a hierarchy portion 354 and a contents portion 358. As is well known to those of ordinary skill in the art, the hierarchy portion 354 provides a hierarchical view of an enterprise, process, and/or plant. Plants, areas within plants, plant components, etc., may be represented, in a hierarchical manner, as folders and subfolders. By selecting a folder in the hierarchy portion 354, with, for example, a mouse, a user can view the contents of that folder in the contents portion 358. In Fig. 11, the folder "Boiler 1" has been selected in the hierarchy portion 354. The folder "Boiler 1" includes six devices as can be seen in the contents portion 358.

[67] In this example, a user has specified, via user interfaces 250, 270, and 290 (Figs. 7-9) that a desired application with the descriptive name "Valve Documentation" is to be associated with various device types including device type "DVC5000." In Fig. 11, the device "FC-4000," which is of the type "DVC5000," has been selected. In this example, the originating application has been configured to include a menu item 366 in a menu associated with devices of the type "DVC5000." The menu 362 can be displayed, for example, by selecting device "FC-4000" with an activation of a left mouse button, and then activating a right mouse button.

[68] As is well known to those of ordinary skill in the art, for applications designed for a WINDOWS™ type operating system, the originating application may be configured to provide one or more user interface mechanisms, such as a menu item, by modifying a configuration file, such as an INI file associated with the origination application, or a "registry" file.

[69] At block 332, the originating application may be configured to launch the application launcher program upon activation of the user interface mechanism (block 328). Additionally, the originating application may be configured to launch the application launcher program with the information formatted at block 324. Continuing with the example described with reference to block 324, the originating application may be configured, when launching the application launcher program, to indicate to the application launcher program the name and location of the file created at block 324. As is well known to those of ordinary skill in the art, for applications designed for a WINDOWS™ type operating system, the originating application may be so configured by modifying a configuration file, such as an INI file associated with the origination application, or a "registry" file.

[70] **Launching the Desired Application**

[71] Fig. 12 is a flow diagram illustrating an example method 400 in a process plant application for launching a desired application. The flow of Fig. 12 will be described with reference to Figs. 11 and 13.

[72] At block 404, a user may activate a user interface mechanism in an originating application. Referring again to Fig. 11, the user may, for example, select a menu item such as menu item 366. At block 408, based on the activated user interface mechanism, the originating application may determine that an application launcher program should be launched. For example, if a menu item is selected by a user, the originating application may determine, based on the selected menu item, that the application launcher program should be launched.

[73] Referring now to Fig. 13, the originating application 450 may examine a first file, registry, etc., or portion of a file, registry, etc., (first file 454) associated with the menu item selected at block 404. The first file 454 may provide the

executable file name and path of the application launcher program, thus indicating that the application launcher program should be launched.

[74] Referring again to Fig. 12, at block 412, the originating application may determine, based on the user interface mechanism activated at block 404, an indicator or indicators of a desired application to be launched and of startup parameters for the desired application. For example, if a menu item is selected by a user, the originating application may determine, based on the selected menu item, an indicator or indicators of a desired application to be launched and of startup parameters for the desired application. In one particular example, the originating application may examine a configuration file or a portion of a configuration file, a registry file, etc., associated with the menu item. This configuration file, registry file, etc., may include a file name and path of another file. This other file may include information regarding the desired application, as well as startup parameters for the desired application.

[75] Referring again to Fig. 13, the originating application 450 may examine the first file 454 associated with the menu item selected at block 404. The first file 454 may provide a file name and path of a second file 458. The second file may include the executable file name and path of the desired application to be launched. Additionally, the second file may include startup parameters for the desired application in a format according to the API of the desired application. For example, the startup parameters may be in a format assembled using the user interface 270 of Fig. 8.

[76] Referring again to Fig. 12, at block 416, the originating application may launch the application launcher program according to the standard API of the originating application (or a non-standard API). Additionally, the originating application may provide, via the API of the application launcher program, the indicator(s) of the desired application to be launched and its startup parameters. In the example of Fig. 13, the originating application 450 may launch the application launcher program 462, and provide it the name and path of the second file 458.

[77] At block 420, the application launcher program may determine the desired application to be launched, based on the indicator(s) provided at block 416. In

the example of Fig. 13, the application launcher program 462 may determine the desired application 466 by examining the second file 458. At block 424, the application launcher program may determine the startup parameters for launching the desired application, based the indicator(s) provided at block 416. In the example of Fig. 13, the application launcher program 462 may determine the startup parameters by examining the second file 458.

[78] At block 428, the application launcher program may launch the desired application according to the API of the desired application. Additionally, the application launcher program may provide startup parameters to the desired application, according to the API of the desired application. In the example of Fig. 13, the application launcher program 462 may launch the desired application specified in the second file 458. Additionally, the application launcher program 462 may provide to the desired application the startup parameters provided in the second file 458. The startup parameters may be formatted in the second file 458 according to the API of the desired application. For instance, a user may use a user interface such as user interface 270 of Fig. 8 to assemble a "command line" that is formatted according to the API of the desired application.

[79] If the startup parameters include arguments (e.g., Device ID, Device Type, etc.), the application launcher program may determine the value of those arguments, and then, for example, may include those values in the command line. In one example, the originating application (or some other application) may pass the values of a set of arguments (e.g., all arguments, all arguments that could possibly be used by the application launcher program, all arguments that it has been determined that the application launcher program will need, etc.) to the application launcher program upon startup. Also, the originating application (or some other application) may store the values of the set of arguments in a file. Then, when the application launcher needs values of particular arguments for launching a desired application, the application launcher program may determine those values by retrieving the needed values from the set of values passed to the application launcher program, or stored in the file. Then, the application launcher program may provide the values of those arguments to the desired application via the API of the desired application.

[80] One of ordinary skill in the art will recognize that an application launcher program may be implemented as a separate program from the originating application, or as a subroutine, object, etc., of the originating application.

[81] The methods described above may be implemented as software programs for execution by one or more processors. Such programs may be embodied in software stored on a tangible medium such as CD-ROM, a floppy disk, a hard drive, a digital versatile disk (DVD), or a memory associated with the processor, but persons of ordinary skill in the art will readily appreciate that the entire program or parts thereof could alternatively be executed by a device other than a processor, and/or embodied in firmware and/or dedicated hardware in a well known manner. Additionally, although the examples described above were described with reference to the flow diagrams of Figs. 4, 6, 10, and 12, persons of ordinary skill in the art will readily appreciate that many other methods of implementing these methods may alternatively be used. For example, the order of execution of the blocks may be changed, and/or the blocks may be changed, eliminated, or combined.

[82] While the invention is susceptible to various modifications and alternative constructions, certain illustrative embodiments thereof have been shown in the drawings and are described in detail herein. It should be understood, however, that there is no intention to limit the disclosure to the specific forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions and equivalents falling within the spirit and scope of the disclosure as defined by the appended claims.

What is Claimed is:

1. A method in a process plant for configuring an originating application to cause a desired application to launch, the method comprising:

prompting a user to indicate at least one desired application;

prompting the user to indicate at least one component, in a process plant, with which the at least one desired application is to be associated;

associating the at least one desired application with the at least one component.

2. A method as defined in claim 1, wherein the at least one component includes a controller.

3. A method as defined in claim 1, wherein the at least one component includes a control strategy.

4. A method as defined in claim 1, wherein the at least one component includes a module.

5. A method as defined in claim 1, wherein the at least one component includes a device.

6. A method as defined in claim 5, wherein prompting the user to indicate at least one component comprises prompting the user to indicate a specific device.

7. A method as defined in claim 5, wherein prompting the user to indicate at least one component comprises prompting the user to indicate device type.

8. A method as defined in claim 1, wherein associating the at least one desired application with the at least one component comprises storing an indication of an association of the at least one desired application with the at least one component.

9. A method as defined in claim 1, wherein associating the at least one desired application with the at least one component comprises storing an indication of the at least one desired application in a file associated with the at least one component.

10. A method as defined in claim 1, wherein associating the at least one desired application with the at least one component comprises storing an indication of the at least one desired application in a portion of a file, the portion of the file associated with the at least one component.

11. A method as defined in claim 1, further comprising:

prompting the user to indicate at least one startup parameter for launching the at least one desired application; and

associating the at least one startup parameter with the at least one desired application and with the at least one component.

12. A method as defined in claim 11, wherein associating the at least one startup parameter with the at least one desired application and with the at least one component comprises storing an indication of an association of the at least one startup parameter with the at least one desired application and with the at least one component.

13. A method as defined in claim 11, wherein associating the at least one startup parameter with the at least one desired application and with the at least one component comprises storing an indication of the at least one startup parameter in a file associated with the at least one component.

14. A method as defined in claim 11, wherein associating the at least one startup parameter with the at least one desired application and with the at least one component comprises storing an indication of the at least one startup parameter in a portion of a file, the portion of the file associated with the at least one component.

15. A method as defined in claim 11, wherein prompting the user to indicate at least one startup parameter comprises prompting the user to input at least one parameter.

16. A method as defined in claim 11, wherein prompting the user to indicate at least one startup parameter comprises prompting the user to select at least one startup parameter from a menu of startup parameters.

17. A method as defined in claim 16, wherein the menu of startup parameters includes arguments utilized by the originating application.

18. A method as defined in claim 11, wherein prompting the user to indicate at least one startup parameter comprises prompting the user to specify an ordering of multiple startup parameters.

19. A method as defined in claim 11, wherein prompting the user to indicate at least one startup parameter comprises prompting the user to specify a command line syntax.

20. A method as defined in claim 1, wherein prompting the user to indicate at least one desired application includes prompting the user to specify a name of an executable file of the at least one desired application.

21. A method as defined in claim 1, wherein prompting the user to indicate at least one desired application includes prompting the user to specify a location of an executable file of the at least one desired application.

22. A method as defined in claim 1, further comprising modifying an originating application to include a user interface mechanism for launching the at least one desired application, the user interface mechanism associated with the at least one component.

23. A method as defined in claim 22, wherein the user interface mechanism includes an item in a menu, the menu associated with the at least one component.

24. A method as defined in claim 22, wherein modifying the originating application includes modifying a database that includes configuration information associated with the originating application.

25. A method in a process plant for configuring asset management software to cause an application to launch, the method comprising:

prompting a user to indicate a desired application;

prompting a user to indicate startup parameters to be supplied to the desired application;

prompting the user to indicate a device to which the desired application is to be associated; and

associating the desired application and the startup parameters with the device.

26. A method as defined in claim 25, wherein associating the desired application with the device comprises storing an indication of an association of the desired application with the device.

27. A method as defined in claim 25, wherein associating the desired application with the device comprises storing an indication of the desired application in a file associated with the device.

28. A method as defined in claim 25, wherein associating the desired application with the device comprises storing an indication of the desired application in a portion of a file, the portion of the file associated with the device.

29. A method as defined in claim 25, further comprising modifying the originating application to include a user interface mechanism for launching the desired application, the user interface mechanism associated with the device.

30. A method as defined in claim 25, wherein prompting the user to indicate the device comprises prompting the user to indicate a specific device.

31. A method as defined in claim 25, wherein prompting the user to indicate the device comprises prompting the user to indicate a device type.

32. A method in a process plant for launching an application, the method comprising:

receiving, via an originating application in a process plant application, a user command to launch a desired application;

determining, based on the user command, an indication of the desired application;

retrieving, based on the indication of the desired application, information for launching the desired application according to an application program interface of the desired application; and

launching the desired application via the application program interface of the desired application.

33. A method as defined in claim 32, wherein determining the indication of the desired application includes retrieving, based on the user command, the indication from a file.

34. A method as defined in claim 32, wherein retrieving the information for launching the desired application includes retrieving the information from a file.

35. A method as defined in claim 34, wherein the indication of the desired application comprises the name of the file that includes the information for launching the desired application.

36. A method as defined in claim 32, further comprising:

determining, based on the user command, an indication of startup parameters for the desired application; and

retrieving, based on the indication of the desired application, information for launching the desired application with the startup;

wherein launching the desired application comprises providing the startup parameters to the desired application via the application program interface of the desired application.

37. A method as defined in claim 36, wherein the indication of the desired application and the indication of startup parameters are a single indicator.

38. A method as defined in claim 37, wherein the single indicator comprises a name of a file, wherein the file includes a name of an executable file for the desired application and the startup parameters.

39. A method as defined in claim 32, further comprising:

providing the indication of the desired application to an application launcher program via an application program interface of the application launcher program;

wherein the application program interface of the application launcher program is different than the application program interface of the desired program;

wherein the application launcher program retrieves the information for launching the desired application;

wherein the application launcher program launches the desired application via the application program interface of the desired application.

40. A method in a process plant for launching an application, the method comprising:

receiving, via an originating application in a management system application, a user command associated with a device in a process plant;

determining, based on the user command, an indication of a desired application;

determining, based on the user command, an indication of startup parameters for the desired application;

retrieving, based on the indication of the desired application, information for launching the desired application according to an application program interface of the desired application;

retrieving, based on the indication of the startup parameters, information for providing the startup parameters to the desired application via the application program interface of the desired application;

launching the desired application, with the startup parameters, via the application program interface of the desired application.

41. A tangible medium storing machine readable instructions, comprising:

first software to prompt a user to indicate at least one desired application;

second software to prompt the user to indicate at least one component, in a process plant, with which the at least one desired application is to be associated;

third software to associate the at least one desired application with the at least one component.

42. A tangible medium storing machine readable instructions, comprising:

first software to receive, via an originating application in a process plant application, a user command to launch a desired application;

second software to determine, based on the user command, an indication of the desired application;

third software to retrieve, based on the indication of the desired application, information for launching the desired application according to an application program interface of the desired application; and

fourth software to launch the desired application via the application program interface of the desired application.

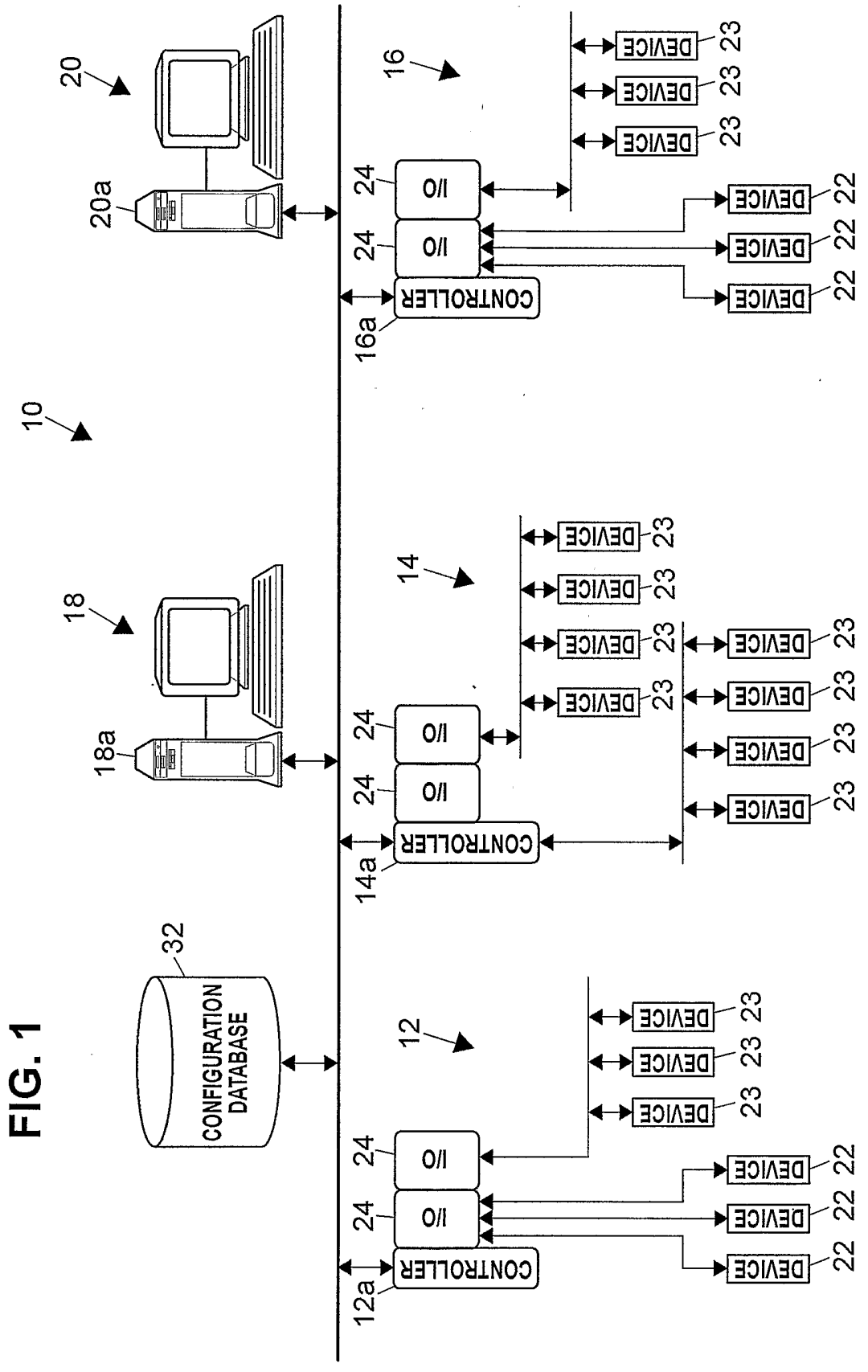

10
12
12a
14
14a
16
16a
18
18a
20
20a
32
CONFIGURATION DATABASE
CONTROLLER
I/O
DEVICE
22
23
24

FIG. 1

FIG. 2

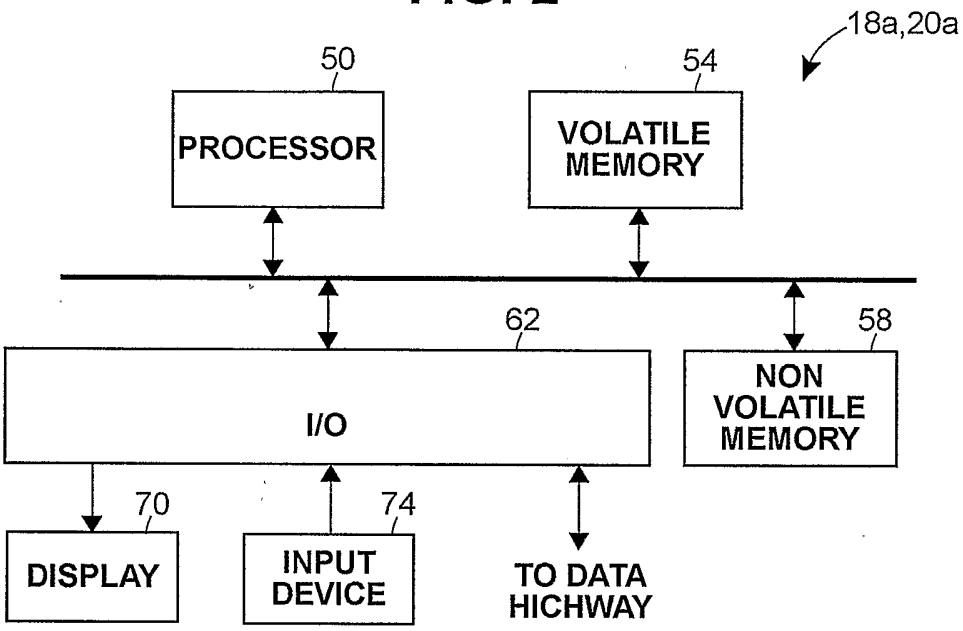

18a,20a
50
54
PROCESSOR
VOLATILE MEMORY
62
58
I/O
NON VOLATILE MEMORY
70
74
DISPLAY
INPUT DEVICE
TO DATA HICHWAY

FIG. 3

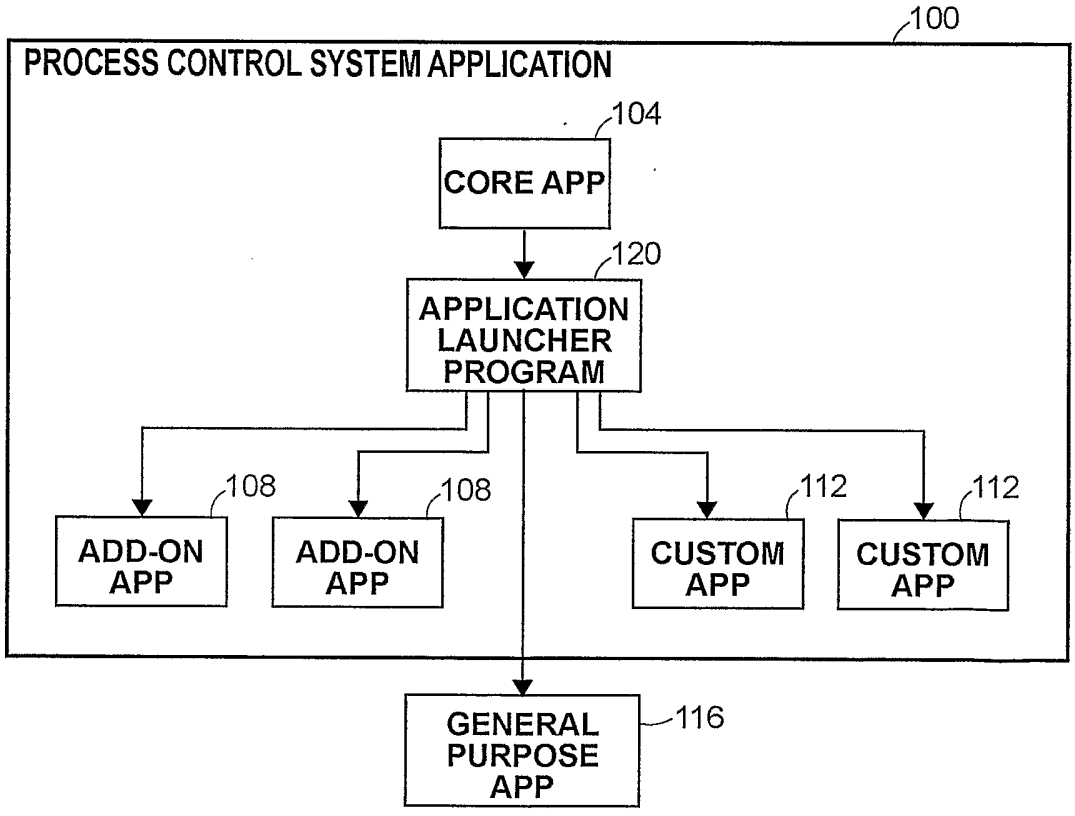

100
PROCESS CONTROL SYSTEM APPLICATION
104
CORE APP
120
APPLICATION LAUNCHER PROGRAM
108
108
112
112
ADD-ON APP
ADD-ON APP
CUSTOM APP
CUSTOM APP
GENERAL PURPOSE APP
116

FIG. 4

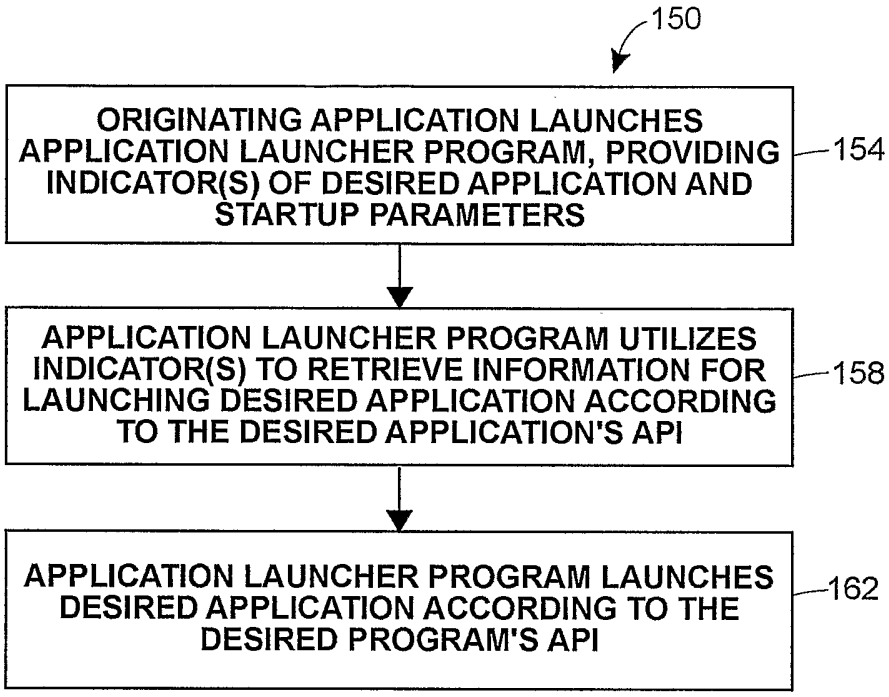

150
ORIGINATING APPLICATION LAUNCHES APPLICATION LAUNCHER PROGRAM, PROVIDING INDICATOR(S) OF DESIRED APPLICATION AND STARTUP PARAMETERS
154
APPLICATION LAUNCHER PROGRAM UTILIZES INDICATOR(S) TO RETRIEVE INFORMATION FOR LAUNCHING DESIRED APPLICATION ACCORDING TO THE DESIRED APPLICATION'S API
158
APPLICATION LAUNCHER PROGRAM LAUNCHES DESIRED APPLICATION ACCORDING TO THE DESIRED PROGRAM'S API
162

FIG. 5

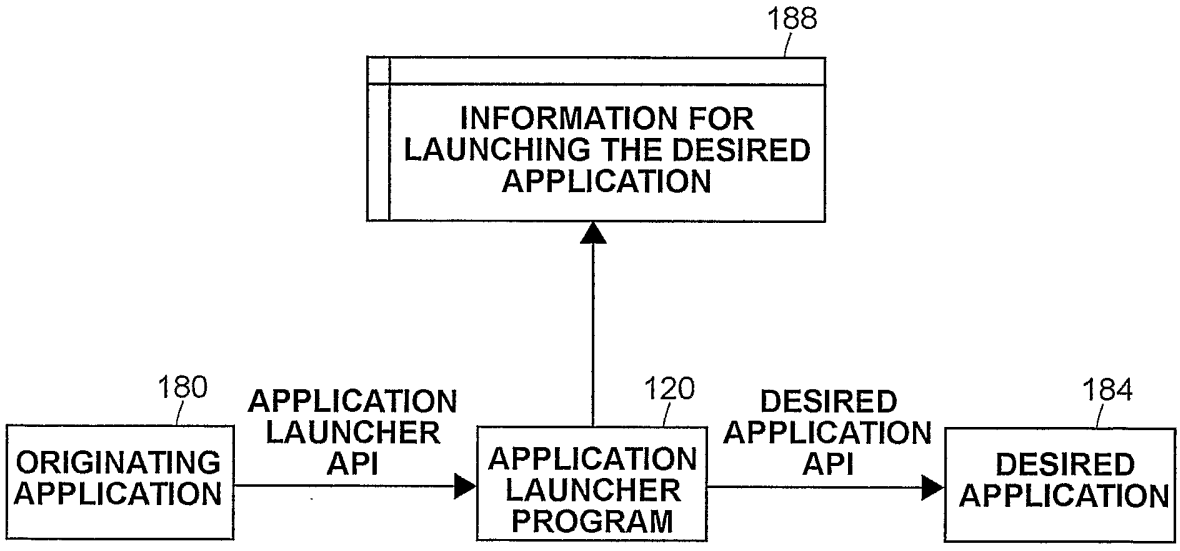

188
INFORMATION FOR LAUNCHING THE DESIRED APPLICATION
180
ORIGINATING APPLICATION
APPLICATION LAUNCHER API
120
APPLICATION LAUNCHER PROGRAM
DESIRED APPLICATION API
184
DESIRED APPLICATION

FIG. 6

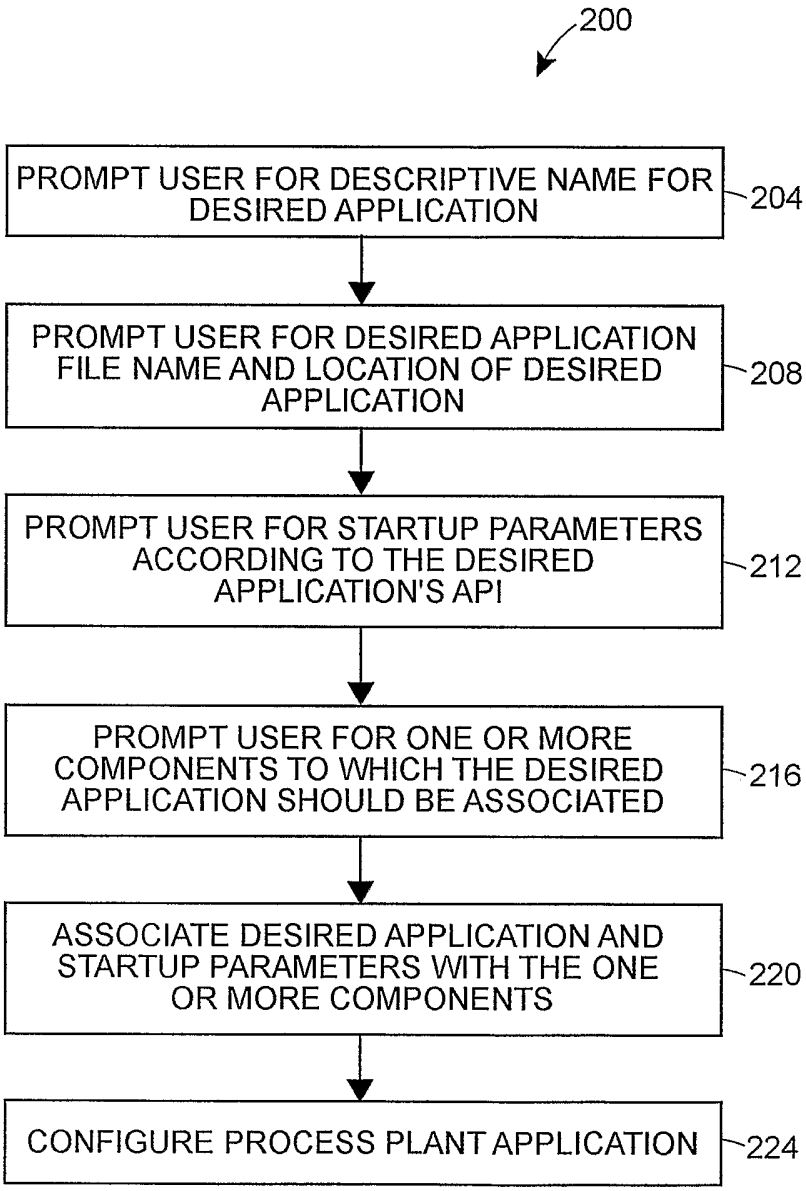

200
PROMPT USER FOR DESCRIPTIVE NAME FOR DESIRED APPLICATION
204
PROMPT USER FOR DESIRED APPLICATION FILE NAME AND LOCATION OF DESIRED APPLICATION
208
PROMPT USER FOR STARTUP PARAMETERS ACCORDING TO THE DESIRED APPLICATION'S API
212
PROMPT USER FOR ONE OR MORE COMPONENTS TO WHICH THE DESIRED APPLICATION SHOULD BE ASSOCIATED
216
ASSOCIATE DESIRED APPLICATION AND STARTUP PARAMETERS WITH THE ONE OR MORE COMPONENTS
220
CONFIGURE PROCESS PLANT APPLICATION
224

FIG. 7

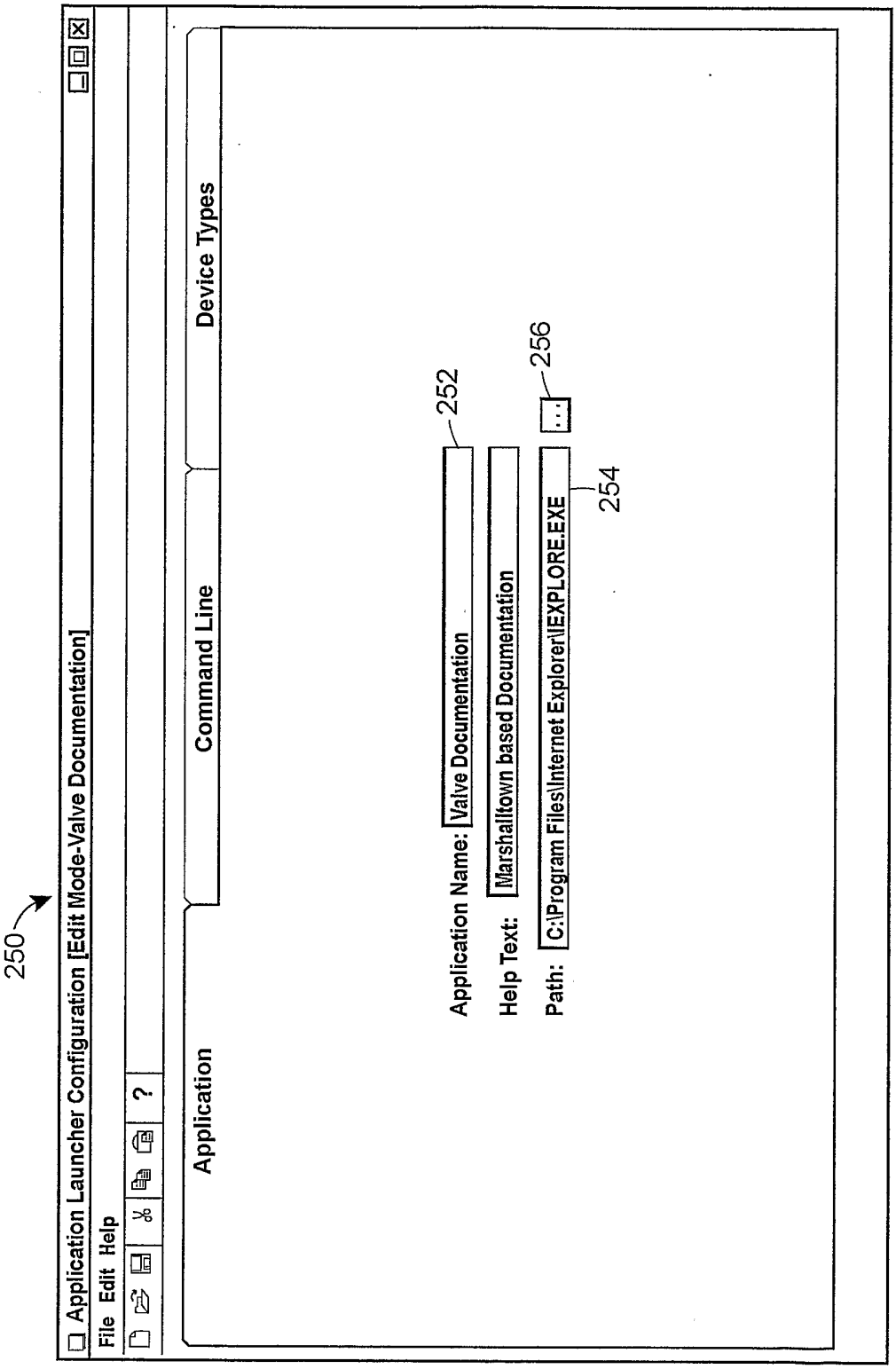

250
Application Launcher Configuration [Edit Mode-Valve Documentation]
File Edit Help
?
Application
Command Line
Device Types
Application Name: Valve Documentation
252
Help Text: Marshalltown based Documentation
Path: C:\Program Files\Internet Explorer\IEXPLORE.EXE
254
256

FIG. 8

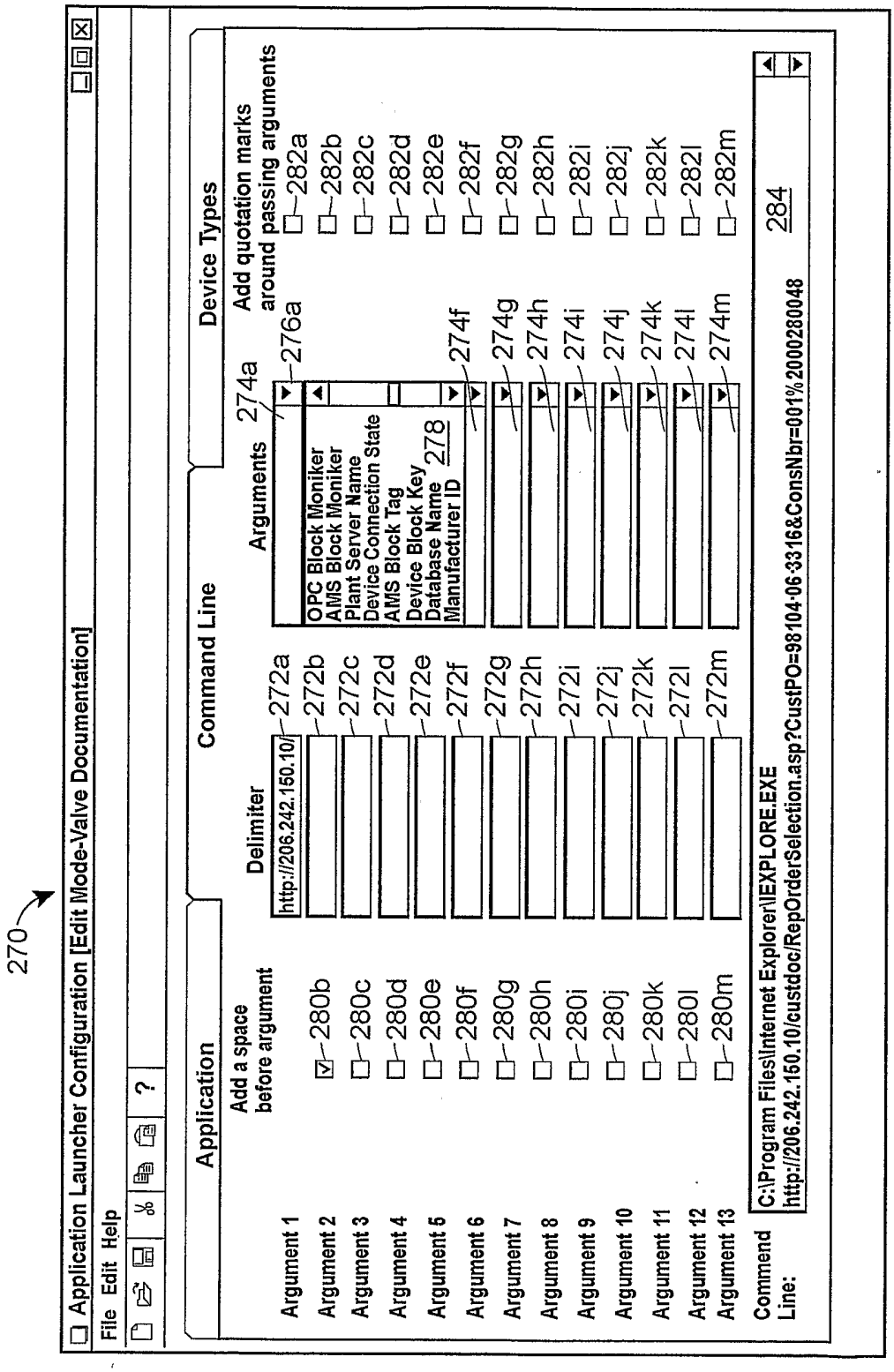

270
Application Launcher Configuration [Edit Mode-Valve Documentation]
File Edit Help
Application
Command Line
Device Types
Add a space before argument
Delimiter
http://206.242.150.10/
Arguments
274a
276a
Add quotation marks around passing arguments
OPC Block Moniker
AMS Block Moniker
Plant Server Name
Device Connection State
AMS Block Tag
Device Block Key
Database Name
Manufacturer ID
278
Argument 1
Argument 2
Argument 3
Argument 4
Argument 5
Argument 6
Argument 7
Argument 8
Argument 9
Argument 10
Argument 11
Argument 12
Argument 13
280b
280c
280d
280e
280f
280g
280h
280i
280j
280k
280l
280m
272a
272b
272c
272d
272e
272f
272g
272h
272i
272j
272k
272l
272m
274f
274g
274h
274i
274j
274k
274l
274m
282a
282b
282c
282d
282e
282f
282g
282h
282i
282j
282k
282l
282m
Command Line:
C:\Program Files\Internet Explorer\IEXPLORE.EXE http://206.242.150.10/custdoc/RepOrderSelection.asp?CustPO=98104-06-3316&ConsNbr=001%200028048
284

FIG. 9

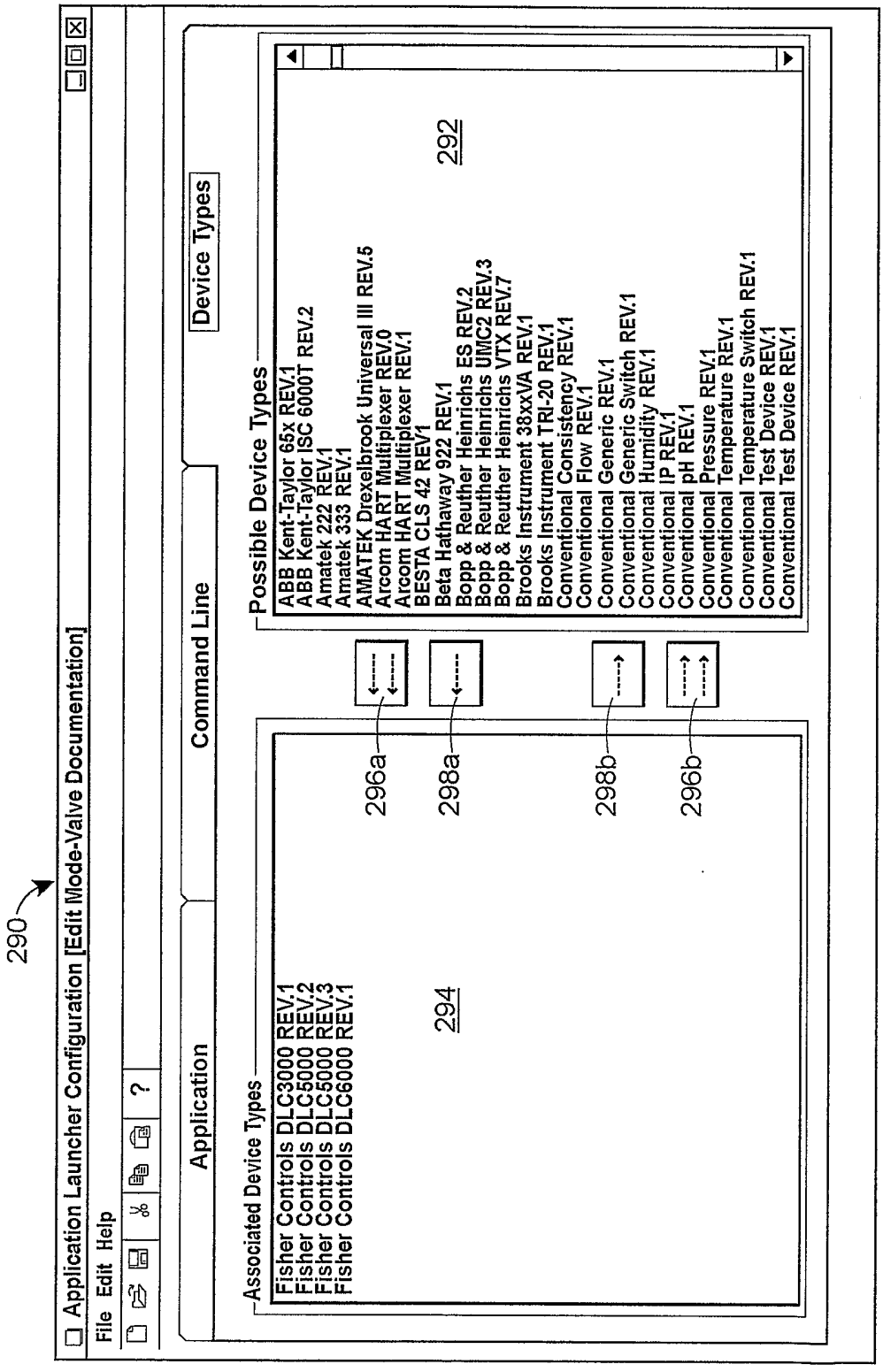

290
Application Launcher Configuration [Edit Mode-Valve Documentation]
File Edit Help
Application
Command Line
Device Types
Associated Device Types
Fisher Controls DLC3000 REV.1
Fisher Controls DLC5000 REV.2
Fisher Controls DLC5000 REV.3
Fisher Controls DLC6000 REV.1
294
296a
298a
298b
296b
Possible Device Types
292
ABB Kent-Taylor 65x REV.1
ABB Kent-Taylor ISC 6000T REV.2
Amatek 222 REV.1
Amatek 333 REV.1
AMATEK Drexelbrook Universal III REV.5
Arcom HART Multiplexer REV.0
Arcom HART Multiplexer REV.1
BESTA CLS 42 REV.1
Beta Hathaway 922 REV.1
Bopp & Reuther Heinrichs ES REV.2
Bopp & Reuther Heinrichs UMC2 REV.3
Bopp & Reuther Heinrichs VTX REV.7
Brooks Instrument 38xxVA REV.1
Brooks Instrument TRI-20 REV.1
Conventional Consistency REV.1
Conventional Flow REV.1
Conventional Generic REV.1
Conventional Generic Switch REV.1
Conventional Humidity REV.1
Conventional IP REV.1
Conventional pH REV.1
Conventional Pressure REV.1
Conventional Temperature REV.1
Conventional Temperature Switch REV.1
Conventional Test Device REV.1
Conventional Test Device REV.1

FIG. 10

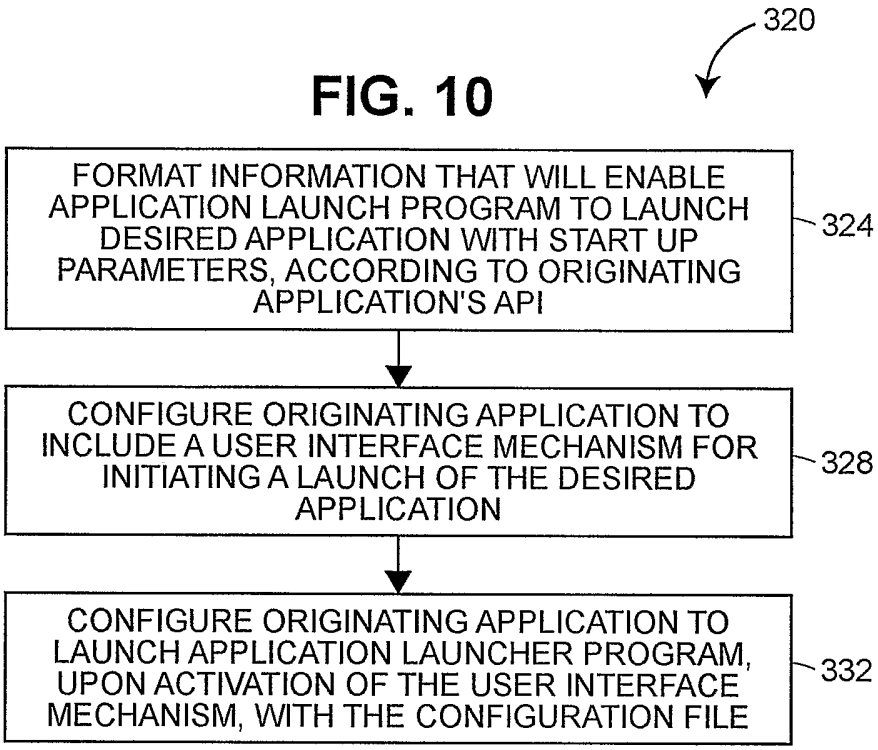
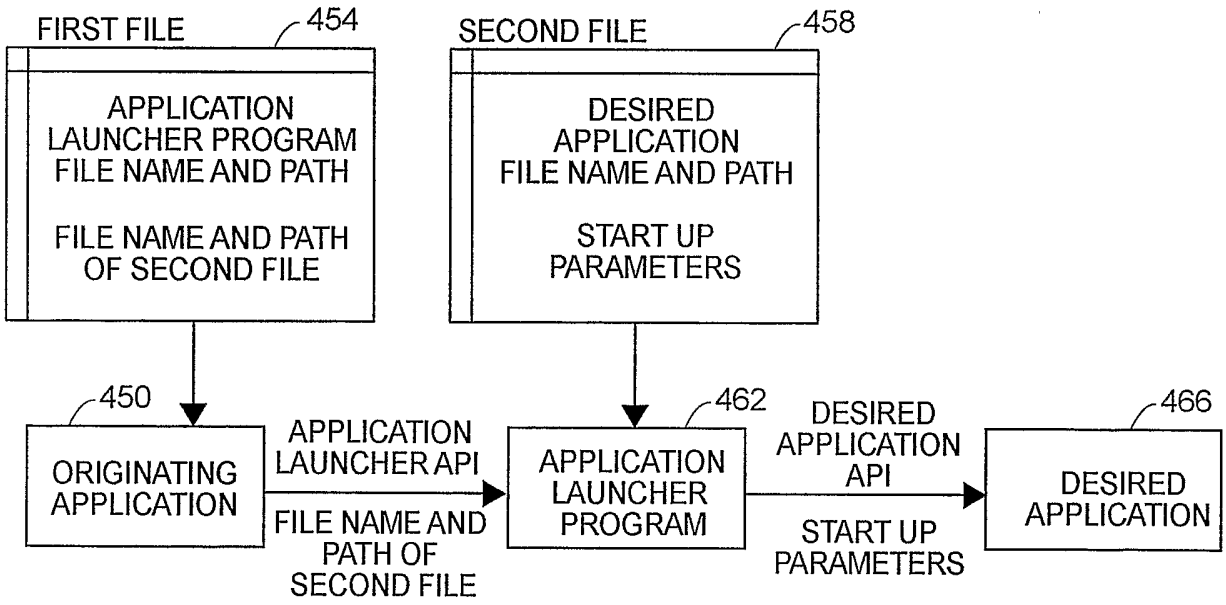

320
FORMAT INFORMATION THAT WILL ENABLE APPLICATION LAUNCH PROGRAM TO LAUNCH DESIRED APPLICATION WITH START UP PARAMETERS, ACCORDING TO ORIGINATING APPLICATION'S API
324
CONFIGURE ORIGINATING APPLICATION TO INCLUDE A USER INTERFACE MECHANISM FOR INITIATING A LAUNCH OF THE DESIRED APPLICATION
328
CONFIGURE ORIGINATING APPLICATION TO LAUNCH APPLICATION LAUNCHER PROGRAM, UPON ACTIVATION OF THE USER INTERFACE MECHANISM, WITH THE CONFIGURATION FILE
332

FIG. 13


FIRST FILE
454
APPLICATION LAUNCHER PROGRAM FILE NAME AND PATH
FILE NAME AND PATH OF SECOND FILE
SECOND FILE
458
DESIRED APPLICATION FILE NAME AND PATH
START UP PARAMETERS
450
ORIGINATING APPLICATION
APPLICATION LAUNCHER API
FILE NAME AND PATH OF SECOND FILE
462
APPLICATION LAUNCHER PROGRAM
DESIRED APPLICATION API
START UP PARAMETERS
466
DESIRED APPLICATION

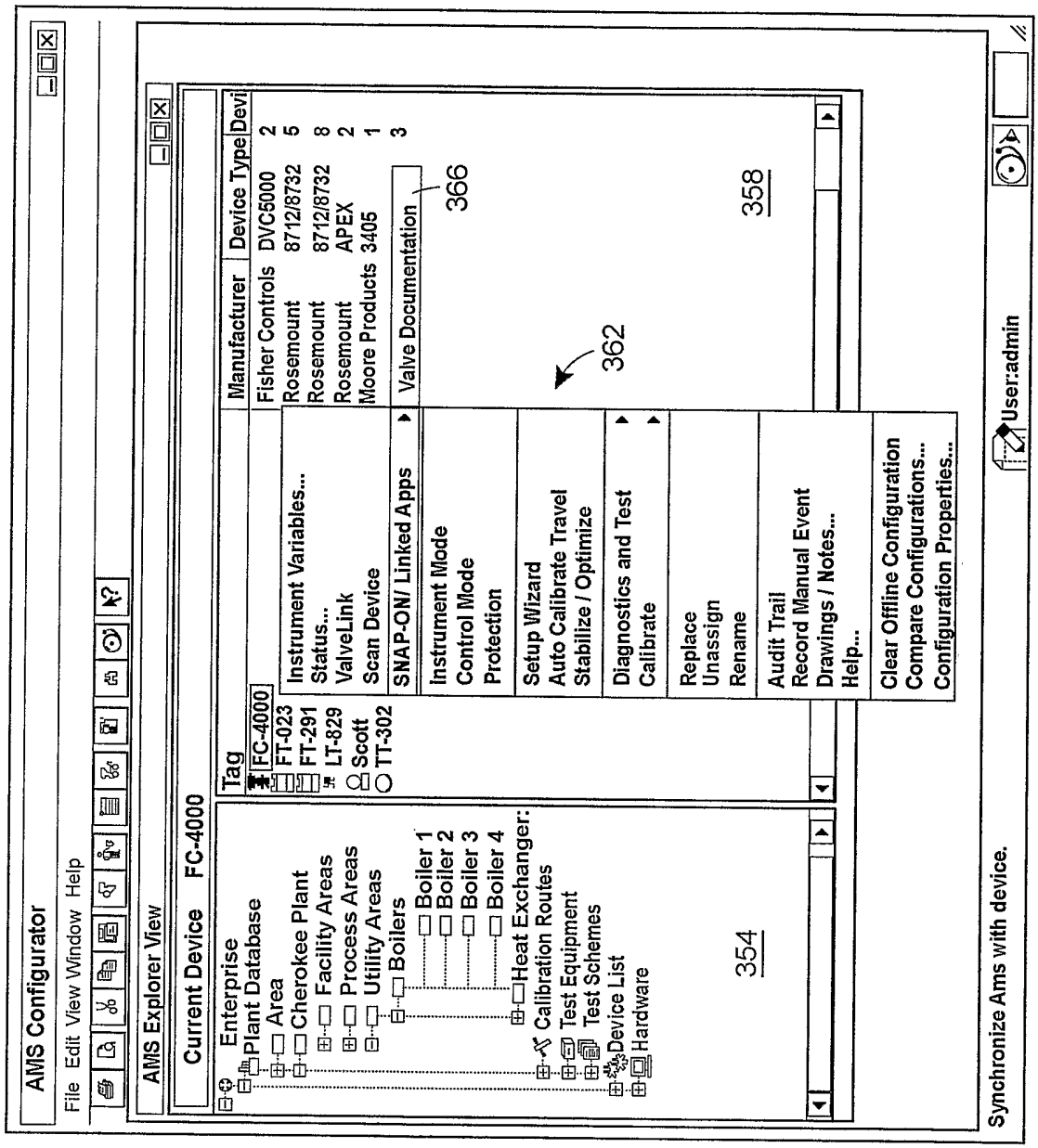
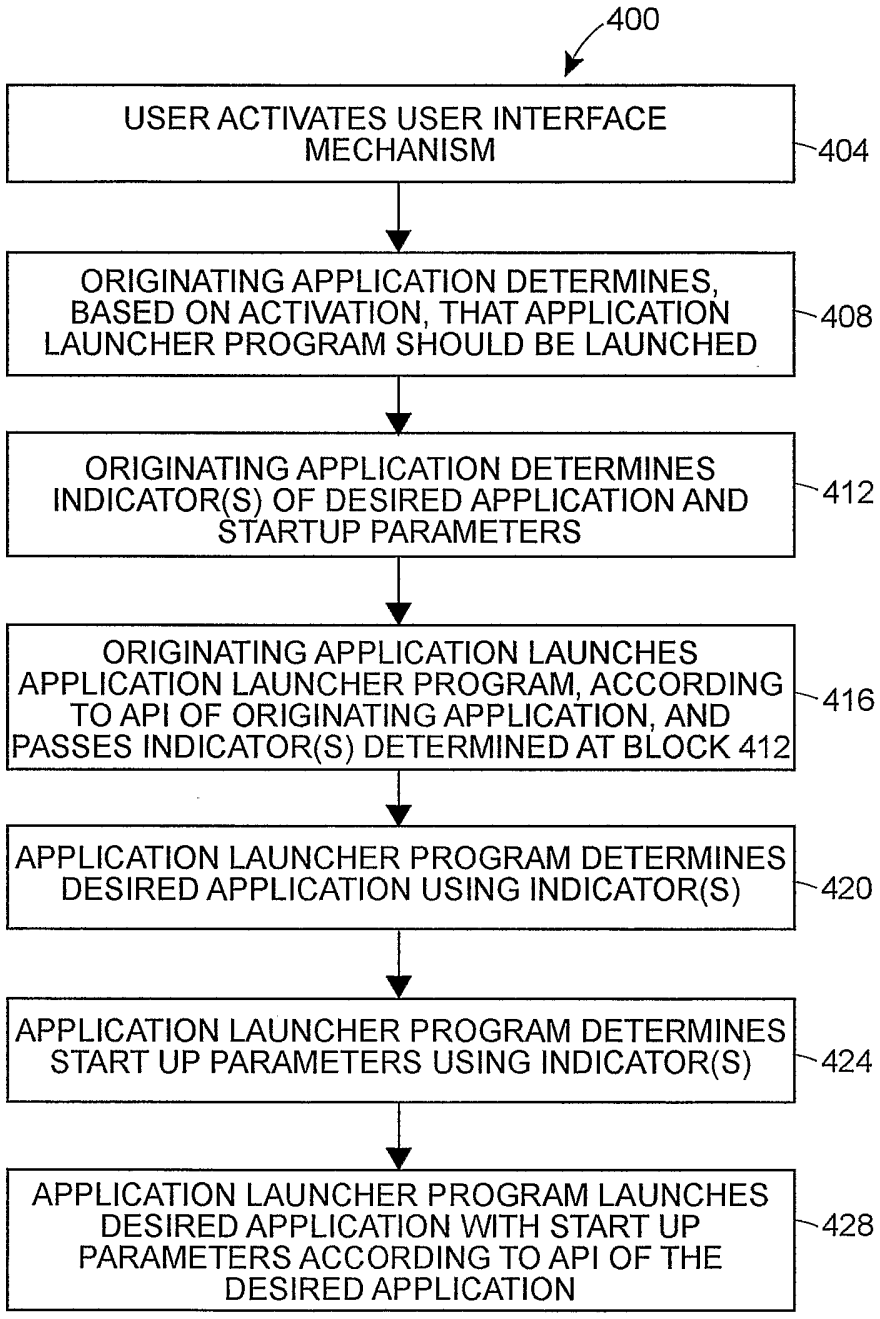

AMS Configurator
File Edit View Window Help
AMS Explorer View
Current Device FC-4000
Enterprise
Plant Database
Area
Cherokee Plant
Facility Areas
Process Areas
Utility Areas
Boilers
Boiler 1
Boiler 2
Boiler 3
Boiler 4
Heat Exchanger:
Calibration Routes
Test Equipment
Test Schemes
Device List
Hardware
354
Tag
FC-4000
FT-023
FT-291
LT-829
Scott
TT-302
Manufacturer
Fisher Controls
Rosemount
Rosemount
Rosemount
Moore Products
Device Type
DVC5000
8712/8732
8712/8732
APEX
3405
Devi
2
5
8
2
1
3
Instrument Variables...
Status...
ValveLink
Scan Device
SNAP-ON/ Linked Apps
Valve Documentation
366
Instrument Mode
Control Mode
Protection
Setup Wizard
Auto Calibrate Travel
Stabilize / Optimize
362
Diagnostics and Test
Calibrate
Replace
Unassign
Rename
Audit Trail
Record Manual Event
Drawings / Notes...
Help...
358
Clear Offline Configuration
Compare Configurations...
Configuration Properties...
User:admin
Synchronize Ams with device.

FIG. 11

FIG. 12


400
USER ACTIVATES USER INTERFACE MECHANISM
404
ORIGINATING APPLICATION DETERMINES, BASED ON ACTIVATION, THAT APPLICATION LAUNCHER PROGRAM SHOULD BE LAUNCHED
408
ORIGINATING APPLICATION DETERMINES INDICATOR(S) OF DESIRED APPLICATION AND STARTUP PARAMETERS
412
ORIGINATING APPLICATION LAUNCHES APPLICATION LAUNCHER PROGRAM, ACCORDING TO API OF ORIGINATING APPLICATION, AND PASSES INDICATOR(S) DETERMINED AT BLOCK 412
416
APPLICATION LAUNCHER PROGRAM DETERMINES DESIRED APPLICATION USING INDICATOR(S)
420
APPLICATION LAUNCHER PROGRAM DETERMINES START UP PARAMETERS USING INDICATOR(S)
424
APPLICATION LAUNCHER PROGRAM LAUNCHES DESIRED APPLICATION WITH START UP PARAMETERS ACCORDING TO API OF THE DESIRED APPLICATION
428