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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 December 2016 (29.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/209993 A1

(51) International Patent Classification:
G06F 17/00 (2006.01)

(21) International Application Number:
PCT/US2016/038809

(22) International Filing Date:
22 June 2016 (22.06.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/183,097 22 June 2015 (22.06.2015) US

(71) Applicant: **INVOTAS CYBER SOLUTIONS, INC.** [US/US]; 201 N. Union Street, Alexandria, Virginia 22314 (US).

(72) Inventors: **BAILEY, Christopher Nelson**; 2792 Adams Street, Adamstown, Maryland 21710 (US). **CONSTANT, Bernd**; 1530 Key Blvd. #1232, Arlington, Virginia 22209 (US). **VELA, Juan Manuel**; 13526 Persian Ct., Woodbridge, Virginia 22193 (US).

(74) Agents: **KONG, Xianhua** et al.; Cooley LLP, 1299 Pennsylvania Avenue, NW, Suite 700, Washington, District of Columbia 20004 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: GRAPHICAL USER INTERFACE ENVIRONMENT FOR CREATING THREAT RESPONSE COURSES OF ACTION FOR COMPUTER NETWORKS

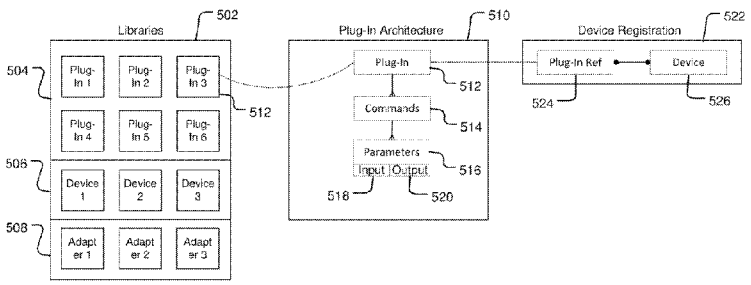


Fig. 5

(57) Abstract: A graphical user interface provides network security administrators a tool to quickly and easily create one or more courses of action for automatic response to a network threat. The courses of action are hardware and system agnostic, which allows a common response task to be implemented by an underlying response engine for any or multiple similar-function devices regardless of brand or version. The course of action builder allows the administrator to use a simple, graphic-based, business modeling concept to craft and design security response processes rather than having to hard code response routines specific to each piece of hardware on the network. The graphic interface model allows the user of the threat response software incorporating the course of action builder to easily understand the overall flow and paths the response may take, as well as understand the data requirements and dependencies that will be evaluated.

WO 2016/209993 A1

GRAPHICAL USER INTERFACE ENVIRONMENT FOR CREATING THREAT RESPONSE COURSES OF ACTION FOR COMPUTER NETWORKS

Cross-Reference to Related Application

This application claims priority to and the benefit of U.S. Provisional Application No. 62/183,097, filed June 22, 2015 and titled Graphical User Interface Environment For Creating Threat Response Courses of Action for Computer Networks," which is incorporated herein by reference in its entirety.

Technical Field

The technology described herein relates to the creation of threat response actions within a computer network and, more particularly, to the use of a graphical user interface in the form of process workflow representation tools that are used to build response actions.

Background

Cyber-security detection and response systems may be configured to aggregate and unify data from multiple devices, components, and platforms on a computer network. Security administrators often design and implement a standard operating procedure of device-actions taken by security individuals in response to a security incident. Based on the nature of a particular threat, the cyber-security system may initiate an action plan that is tailored to the security operations center and its operating procedures to protect potentially impacted components and network resources. The goal of cyber-security systems is to provide rapid and reliable, enterprise-wide threat responses, e.g., to mitigate threats, survive breaches, and maintain operations during attacks. To provide rapid responses, security administrators often program preconfigured response plans for implementation upon recognition of a cyber-security threat. The cyber-security system can thus provide system configuration instructions to defend against threats originating both external to and internal to the network. At present such specific threat response plans and related system configuration instructions must be individually coded by the security administrator. The coding in the response plan must also be specific to the hardware on the network of the enterprise to activate, deactivate, or otherwise reconfigure the hardware and other network systems to respond to the particular security threat identified.

The information included in this Background section of the specification, including any references cited herein and any description or discussion thereof, is included for technical reference purposes only and is not to be regarded subject matter by which the scope of the invention as defined in the claims is to be bound.

Summary

The technology disclosed herein provides a graphical user interface through which users, generally network security administrators, can quickly and easily create one or more courses of action for automatic response to a network threat. The courses of action are hardware and system agnostic, which allows a common response task to be implemented by an underlying response engine for any or multiple similar-function devices regardless of brand or version. The course of action builder allows the administrator to use a simple, graphic-based, business modeling concept to craft and design security response processes rather than having to hard code response routines specific to each piece of hardware on the network. The graphic interface model allows the user of the threat response software incorporating the course of action builder to easily understand the overall flow and paths the response may take, as well as understand the data requirements and dependencies that will be evaluated.

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. A more extensive presentation of features, details, utilities, and advantages of the present invention as defined in the claims is provided in the following written description of various embodiments and implementations and illustrated in the accompanying drawings.

Brief Description of the Drawings

Fig. 1 is a schematic diagram of an exemplary course of action representation built in a graphical user interface for creating automated network security threat response action protocols.

Fig. 2 is a flow diagram of a process implemented through a graphical user interface for building a course of action for an automated network security threat response.

Fig. 3 is an exemplary screenshot of a plugin interface within a course of action builder for selection of available network device plugins and registration with network devices.

Fig. 4 is an exemplary screenshot of an interface for defining input parameter values specific to a command implemented by a network device plug-in.

Fig. 5 is a schematic diagram of a process for registration of network device interface plugins with network devices.

Fig. 6A is an exemplary screenshot of a workflow palate provided by a graphical user interface in a course of action builder with an array of stencil icons.

Fig. 6B is an exemplary screenshot of a workflow palate provided by a graphical user interface in a course of action builder with a start node and an initial task

Fig. 7 is an exemplary screenshot of a task configuration window opening within the workflow plate to provide an interface for configuring the task.

Fig. 8 is an exemplary screenshot of additional layers of the task configuration window of Fig. 7.

Fig. 9A is an exemplary screenshot of a parameter selection element of the task configuration window of Fig. 7.

Fig. 9B is an exemplary screenshot of a completed parameter selection of the task configuration window of Fig. 7.

Fig. 10 is an exemplary course of action workflow created with the course of action builder graphical user interface.

Fig. 11 is schematic diagram of a special purpose computer configured to implement a course of action builder module for building a course of action for an automated network security threat response.

Detailed Description

A cyber-security response system is provided that allows a user to agnostically configure appropriate threat responses by multiple devices, components, and platforms on a computer network. Security administrators can utilize the solution to design and implement a workflow of device-actions to be automatically implemented in response to a security incident. Based on the nature of a particular threat, the cyber-security response system may initiate an action plan that is tailored to the security operations center (SOC) and its operating procedures to protect potentially impacted components and network resources. Upon recognition of a cyber-security threat, through preconfigured activation workflow plans, the

cyber-security system automatically provides system configuration instructions to defend against threats originating both external to and internal to the network.

Users can create courses of action (CoA) in cyber security terms using a CoA Builder module that leverages representational elements in a graphical user interface (GUI). A course of action can be represented in the GUI as a workflow or a sequence of tasks can be generated by the workflow. A CoA response may be built via an intuitive GUI that provides drag and drop base elements in combination with configuration features that transform the graphic elements into specific command or task representations linked to implementable instructions to control devices on the network. Particular network devices are identified during build-out of the CoA and are associated with device plug-ins. The device plug-ins are built to interact with the specific application program interface (API) of the network device and are configured with specific activation routines that can activate or configure the network device in ways that are helpful to blocking or remediating a network security threat. The device plug-ins receive device configuration information from the CoA, identify input parameter information necessary to control the particular device to perform a specific task or action, and identify output information generated by the device when performing the action. Once built in the GUI, a CoA visually depicts the flow of tasks (also referred to herein as "workflow") that can occur when a particular cyber security event occurs and allows users to manage the event response.

The tasks represented in the CoA are implemented in response to a threat event by a separate but integrated application layer of the cyber-security response system referred to herein as the "Orchestration Engine." The Orchestration Engine executes the instructions identified by the CoA when processing inbound events to automatically remediate threats. The Orchestration Engine reads the tasks in the CoA, calls on the particular plug-ins associated with network devices identified in the CoA task to provide device-specific action instructions, passes the necessary input and output parameters to and from the CoA, the plug-ins, and the devices, and processes the CoA workflow in the designed order.

As noted, each CoA is device and platform agnostic. For example, the CoA may be represented in extensible markup language (XML) and the XML file for a CoA defining a response to a threat can be shared from one organization to another that also have a compatible cyber-security response system. While the response management system needs to be compatible, the network configuration and devices thereon can be entirely different between organizations, and the CoA can still work. For example, one organization may have two firewall components while a second organization has ten firewall components. The

firewall devices may be made by different manufacturers. The CoA can generically state that the task is to activate or configure a firewall in a certain way. The underlying Orchestration Engine can thus recognize which device plug-ins correspond to firewall devices that have been registered with the cyber-security response system and then use the values in the generic input parameters received from the CoA to configure the plug-ins to actuate the devices and manage the threat response.

Organizations can create a library of courses of actions (actionable workflows, sequence of actions, or sequence of tasks) so processes are centrally managed and executed in a high-confidence, repeatable fashion via a graphical user interface. Such a library may be maintained as a repository of automated response procedures that can be updated at any time. This gives organizations a single place to go to update a process where it can take effect with zero variation due to it being automated. Any CoAs within a library can be consumed by the Orchestration Engine and automatically executed as soon as they are updated. This leaves zero lag time between the implementation or update of a process and the execution of it. A benefit of a CoA library is that effective threat response processes can be stored to reduce loss of human created know-how when human resources leave. This allows organizations to maintain up-to-date processes in the ever changing landscape of cyber threats. Normalization of data and presentation of threat response processes via a GUI to the user in easily understandable terms allows for ease of management and deployment of threat response actions and reduction in time spent setting up processes. The CoA Builder GUI also allows for more expedient onboarding of new human resources as they can understand the processes faster.

An exemplary, generic course of action workflow for a threat response that may be designed in the CoA Builder module is depicted in Fig. 1. The CoA 100 is built using a graphical user interface. The purpose is to detail the sequence of tasks necessary for a threat response and make references to the plug-in and calls to its components needed for the desired device action. The CoA may be stored as a JavaScript Object Notation (JSON) object or other similar data-interchange format. The CoA does not store code, but instead stores references to the plug-in data and maintains the sequence, order, and conditions by which each task is executed. The CoA 100 also records what input parameters are available based on the tasks chosen and displays the possible output parameters of those tasks based on the existing references to the plug-in data.

The start node 102 of any given CoA 100 is intended to serve as a trigger event. The trigger event is used to create context around the CoA 100 since the sequence of steps is in

response to the inbound event. The start node 102 can ask the user to select an "adapter." The adapter could be anything that would serve as the source of the event requiring response, such as an email, or a sys.log message from a security information and event management (SIEM) application. The user can create a trigger condition associated with the start node 102, which is an evaluation of the incoming data. If the condition is met, the CoA 100 can execute. If the condition is not met, the CoA 100 cannot execute. The trigger condition serves as the business justification for execution of the CoA 100. A specific adapter can be the start node 102 of more than one CoA 100.

The second step to configuring the start node 102 is to select the specific output parameters from the adapter that you wish to make available to the downstream tasks in the CoA 100. Since the instantiation of the adapter in a particular CoA 100 is specific to the trigger condition, the user may also select any data elements considered important for the particular CoA 100 to output as output parameters. This can be contextually different for each CoA into which the adapter is incorporated since the instantiation of the particular CoA is specific to the trigger condition. The output parameters can be selected in the CoA building process. The result of this step is the definition of trigger or start conditions and output parameters from the adapter event.

Once the start node 102 is configured, one or more tasks 104 are then configured to build the workflow of the CoA 100. Configuration of a task 104 involves selecting the following: a device plug-In, a command, and input parameters for the command. By selection of the plug-in to perform the task on, the CoA Builder application knows the available devices due to a reference provided in a device table. Plug-in and device registration can be described in detail later herein in association with Figs. 3-5. Using the plug-in registration information, the associated commands are then presented to the user to select. Once the user selects the command to execute, the plug-in registration information is once again called to provide prompts for the user to enter the input parameters into the task 104. The CoA Builder may allow the user to input the required input parameters by either selecting from the available ones output from an upstream adapter or task 104. Alternatively, the CoA Builder may allow the user to create a custom parameter value with operators such as "begins with," "contains," etc. This provides the flexibility to define a parameter not previously pulled in the workflow, or define something "hard coded" rather than variable. Once the input parameters have been selected, the configuration of the task 104 is complete. The output parameters are automatically output and made available to the downstream CoA tasks so they may also consume or evaluate them.

The CoA 100 may additionally include decision points 106, 112 in paths of the flow to determine whether additional tasks in the workflow should be performed or to choose between alternate tasks. Configuration of a decision point 106, 112 works similarly to a task 104, except that a specific plug-in is not input. The decision point 106, 112 is used to perform an evaluation to determine if the path should continue. Configuration of the decision point 106, 112 also utilizes the graphical user interface to allow the user to construct a conditional statement that can be evaluated to be true or false. For example, if a parameter value is "true," the path may continue; alternatively, if a parameter value is "false," the path may not continue. Any data that has been output from start node 102 or task 104 can be available for evaluation.

Sub-processes 108 can also be incorporated in to CoA 100. This provides the ability to maintain a highly repeatable sequence of tasks and add it in to another CoA without having to recreate it multiple times or maintain multiple instantiations of the same sequence. When a sub-CoA 108 is first created, specific input parameters for the sub-CoA 108 are also defined. When the sub-CoA 108 is incorporated in to a parent CoA 100, those input parameters from the parent CoA 100 can be mapped similarly to a task 104 so all the individual tasks within that sub-CoA 08 can run. A sub-CoA 108 can be designed to run either as an automated action sequence or to run when manually initiated by a user. The latter may require an approval step within a separate authorization tool whereby the sub-CoA 108 can be "recommended" to run and the user may choose to activate it by clicking on it.

End points 110, 114 simply define the end of the CoA 100 and signal to the Orchestration Engine that no further tasks need to be performed.

Fig. 2 depicts an exemplary process 200 for allowing a user to configure a CoA using the CoA Builder GUI tools. The steps in Fig. 2 can further be explained with reference to Figs. 3-9. A preliminary step 202 in configuring a CoA is to register and appropriate device plug-ins within the cyber-security response system that correspond to network components or appliances on the network to be protected. As shown in the screenshot of Fig. 3, a library 300 with a section 316 for device plug-ins 302 may be provided as part of CoA Builder platform.

A plug-in 302 (or connector) is an integration to a third party product (device) that is used to either extract information or modify a policy. Plug-ins 302 are code packs that allow the application to communicate with external devices and applications. The plug-ins 302 allow for communication with network devices and appliances to occur with specific protocols and technologies necessary based on the application protocol interface (API)

dictated by the external device. Each third party product has an available library of commands which they make available to call devices via an API.

Plug-ins 302 developed for the CoA Builder module are provided with commands that serve a purpose in cyber security or network management policies. All plug-ins 302 developed for one or more cyber security applications are delivered with at least one command. Plug-in commands generally correlate directly with a "task" that one would see in a standard operating procedure for security response. Therefore, for a plug-in associated with a firewall, a highly utilized command would be to block an IP Address. Different vendors may call the command something different. One may call it "block_IP" and another may call it "BlockIP," whereas another may use a ID number like "8012A". There is no standard naming convention from product to product, or from one vendor to another. When packaged in the CoA Builder platform, the plug-in commands are aliased to a common and easy to understand action, such as Block IP Address. Within the GUI, the alias would be displayed so the user can quickly understand the task and select it. Aliasing of commands is a technique used throughout the CoA Builder so users can quickly understand the content on the screen and proceed with their operation more efficiently and accurately.

A plug-in 302 is often related to a specific product, for example, a Cisco® ASA Firewall, Microsoft® Active Directory, or iSight Threatscape. As shown in Fig. 3, when a plug- in is selected, such as selected plug-in 304, an interface 306 may be provided to configure the selected plug-in 304. Each plug-in can have at least one command 308. As shown in Fig. 3, the commands button 308 is highlighted and a number of specific commands 312 associated with the selected plugin 304 are presented. Commands 308 are functions that can leverage an API to perform some kind of action on the network asset/device/appliance. Commands 308 are also incorporated in to CoA workflows as "tasks" performed within the CoA. Each command 308 may have a predefined set of input parameters 310 that are required for the plug-in command 312 to start/run. Each command can also have a predefined set of output parameters that are the result of the command running.

As shown in Fig. 4, upon selection of a specific command 402 from the list of commands 312, the particular input parameters 404 associated with the selected command 402 can be presented for review in a plug-in registration window 400 and possible population by the user. Alternatively, the user can select a separate parameters button 310 to review all of the parameters associated with the selected plug-in 304 directly, without reference to a particular command 312. Input parameters are tagged with a name or alias, as well as a type which indicates the kind of data required such as number, date, etc.

Some input parameters 408 may require population (e.g., device identification and password parameters), while other parameters 406 may either be populated at this time if the value is static or may instead be populated by output parameter data generated by an upstream task in a CoA. These output data elements can be consumed by tasks downstream in the workflow. The result of one task can become an input to the start of another task. This makes the process of building a CoA very simple. The abstracted parameters create interoperability between disparate devices within the network and security system of an organization.

Organizations may have more than one instance of a particular product or device. These instances can each have discrete configurations based upon, for example, differences in model number, port, IP address, or other network location and specific password information to access the device. Each instance is considered a device which may be registered as further described below and stored in a section of the library 300, accessible through the GUI by selecting the device section tab 318. Similarly, start nodes or "adapters" once configured and saved may be accessed from a section of the library 300 through the GUI by selecting the device section tab 320.

The library 300 and functionality of interface 306 in Figs. 3 and 4 is also shown schematically in Fig. 5. The library 502 may have sections for storage of plug-ins 504, devices 506 (i.e., an instantiation of a plug-in configured for a particular device), and adapters 508 (i.e., specific trigger conditions to be used as start nodes in a CoA). These library sections correspond to the tabbed display windows 316, 318, and 320, respectively, in Fig. 3. Each plug-in 512 in the library 504 has a specific plug-in architecture 510 including one or more commands 514 that the plug-in 512 can instantiate to activate or configure a type of network appliance. The commands 514 are built to interface with the API of the specific network appliance to be controlled by a task in a CoA. Each command 514 has associated parameters 516 including input parameters 518 needed to populate the command 514 to perform and action and output parameters 520 that may be generated by the network device after completion of the tasked action and returned to the plugin through the API for further use in the CoA. The output parameters 520 of one command may be used as input parameters 518 for other commands triggered by downstream tasks in the CoA.

Device specific input parameters 518 may include model number, port, IP address, or other network location. Once a plug-in 512 is configured with information specific to a particular device, a device 526 is then registered and the device registry information 522 is stored in a device section 506 of the library 502. The device registry information 522

includes the device identification parameters and a reference pointer 524 to the plug-in 512 used to control the functionality if that specific device. Note that parameters 416 for commands 414 associated with a particular type of plug-in 512 does not have to be stored with each instantiation of a device 526.

Returning to Fig. 2, once the plug-ins have been configured, a user can begin to develop a CoA using the CoA Builder GUI module. The CoA Builder can visually depict the flow of tasks that occurs, and also account for workflow elements such as the following: trigger and start conditions; branching and decision logic; parallel processing of tasks; e-mail notification with templates; recommended tasks; extended wait tasks; scheduled tasks; assignment, escalation and user workflow; and looping. In some implementations, these exemplary tasks include input and output parameter definitions. Each one can be simplified by displaying parameters when configuring.

The CoA Builder module initially presents a workflow palate with task icons as indicated in operation 204. An exemplary implementation of a workflow palate 600 is presented in Figs. 6A and 6B with a number of stencils (also referred to herein as "task stencils", or "objects") available to a user to build a CoA. As indicated in Fig. 6A, a start node 602 representing a selected adapter from the adapter library is provided on the palate 600. Alternatively, the start node 062 could be blank and configured from the palate stencil by selection. When a configured start node 602 is selected on the palate 600, a number of other stencil options may appear to assist the user in building a course of action. These stencils may include the following: a task stencil 604, a path toll stencil 606, a gateway stencil 608, a delete stencil 610, and an end node stencil 612.

As shown in Fig. 6A, the task stencil 604 is under consideration for selection (e.g., the user's computer mouse or other pointer may be hovering over the stencil). Initial selection of a particular stencil may be indicated by a change in appearance of the selected stencil. For example, as shown, the task stencil 604 is larger than the other stencils and is circumscribed by an annular ring. Upon completion of stencil selection, for example, the task stencil 604, all of the stencils may disappear from the palate 600 and a workflow instantiation of the selected stencil may be automatically connected to the prior workflow element by a path. This is shown in Fig. 6B wherein an instantiation of a task 614 is presented in the GUI linked to the start node 602 by a path 616. The user may drag the task 614 (or any other workflow element) to any position on the screen to best present and visualize the CoA workflow. The path 616 can automatically follow the task 614 regardless of where it is placed and maintain

the connection with the start node 602. A user can independently select the path 616 to change its connection points and associations.

In addition to the task stencil 604, in this exemplary implementation a user can select a gateway 606 to place in the middle of a path to split the path, e.g., to build two or more different tasks to operate in parallel ingesting the same input parameter data. The gateway stencil 608 can also be used to place a gateway to rejoin two independent paths. The path stencil 608 may be used to draw additional flow paths between gateways, tasks, and nodes. The paths may further be conditional, i.e., the paths may be configured with decisions or conditions that can be met before the CoA continues down that path, e.g., a parameter value meets a certain threshold.

The palate 600 may also include an end node stencil 610 that can be used to place an end node in along a particular flow path, e.g., after the completion of a task or of a series of tasks, to indicate the end of the CoA. In addition, the palate 600 may provide a delete stencil 612. Selection of the delete stencil 612 can delete the workflow element selected and immediately preceding the stencil array. For example, in Fig. 6A, the start node 602 is selected, thus causing the stencil array to appear. If the delete stencil 612 is selected at this time, the start node 602 would be deleted from the palate 600. Similarly, as shown in Fig. 7, the task 614 has been selected and the stencil array 704 is presented by the GUI of the CoA Builder module. If the delete stencil were selected from the stencil array 704 at this point, the task 614 would be deleted from the palate 600.

As shown in Fig. 7, the task stencil 614 is selected. In addition to the presentation of the stencil array 704 in the GUI, a task configuration window 702 is presented to associate a plug-in, a command, and necessary input parameters with the task 614. The steps of configuring a task in this manner are indicated in Fig. 2. When the task box 706 is selected in the task configuration window 702 in Fig. 7, the configuration process begins as indicated in Fig. 2 with the provision of plug-in selections in operation 206. This operation is further represented in Fig. 8 which depicts a plug-in selection list 802. As indicated in operation 208 in Fig. 2, an API may be provided to access the command and parameter information directly from the master plug-in application associated with the device selected and pass agnostic field information to the CoA Builder to request selection of appropriate commands and input parameters needed to properly instantiate the plug-in in the context of the CoA.

Plug-Ins are delivered with pre-configured input and output parameter data. This can be modified in the plug-In configuration page described above with respect to Fig. 3 and 4. Parameter definitions associated with the selected plug-ins can determine what data is

actually available for configuration in the CoA Builder. This simplifies the user experience when configuring the CoA because in some situations, only a small portion of data are handled to find the right parameter for the task being configured, while in other situations, other data can be processed. A lot of data may be returned with each task execution and certain elements can be treated as "parameters" available for downstream consumption depending upon the input parameters needed by another plug-in associated with a downstream task in the CoA.

Recall that a plug-in may be associated with more than one device on the network. Therefore, when a plug-in is selected from the plug-in list 802 as indicated in Fig. 8, a further device list 804 of network devices registered with that plug-in is generated from all of the registered device plug-ins stored in the device library as indicated in operation 210 of Fig. 2. As previously noted, in some implementations, the device registrations do not include all of the information related to a plug-in, and can include the information specific to the device needed to identify and access the device. Further recall that the plug-ins may include multiple different command options and that each of the options may require different input parameters. Thus, as indicated in operation 212 of Fig. 2, the CoA Builder provides a command list 806 of commands associated with the plug-in for the selected device as indicated in Fig. 8. Once a command is selected from the command list 806, the task box 706 is populated by a shorthand command string 808 of the selected plug-in and command for the task 614.

Based on the command selected, the CoA Builder can provide a selection of input parameters available to execute the command as indicated in operation 214 of Fig. 2 and graphically represented in the GUI of Fig. 9A in which the parameters section 708 of the task configuration window 702 becomes active. The input parameters are specific to the command, and predefined at the plug-in command configuration level. The user may select the parameter values to input. As indicated in operation 216 of Fig. 2 there may be options for selection of a parameter value. Specifically, in this exemplary implementation, the input parameter for the present command may be selected from output parameters generated from tasks previously executed upstream in the workflow or the values may be custom input by the user. These options are indicated in the flow diagram of Fig. 2 and graphically presented in the GUI presentations of Figs. 9A and 9B.

The CoA Builder abstracts, categorizes, tags, and filters all the potential parameters within the instant CoA that could be selected for each input parameter to the new command as provided in operation 218 of Fig. 2. Pertinent parameters can be available for selection;

other data can be filtered out as indicated in operation 220. When mapping tasks within the CoA Builder, each subsequent task can request input data (a parameter) that it can use to process its related command. Without this specification much control and flexibility to the system is lost. When executing a task or plug-in command, s some of the data can be used in decision making and downstream task management, while other data will not be so used. Most of the data is erroneous to the processing needs of the CoA. For example, a task to pull a reputation of an IP address, to then make a decision based on that score may be executed. The decision may be to permanently block the IP address or put it on a temporary block. Embodiments here describe a sophisticated technique of allowing the configuration to occur, while keeping it simple to the user.

The CoA Builder interface can, in some situations, make data elements available as input parameters that are already predetermined to be essential to consumption by the remainder of the flow. It does not make all the data returned via a command available for selection because most of it is erroneous. This subsequent experience simplifies the design achieved with the CoA Builder GUI and benefits the user. Users do not need to sift through rows of data to pick out the important data and map it to the next step. In some cases there can be one option and the single parameter can be pre-selected as an input to a task. In other cases there can be more than one option, and the list for selection as input can represent a small percentage of the overall data transmitted through the entire CoA.

The screenshot in Fig. 9A illustrates this functionality. The lookup box 906a under the parameters section 708 has been limited to parameter values having IP addresses. This can be the type of input parameter used for the command string 808 in the task box 706. As shown, the available parameters for the previously selected command based upon all output parameters previously generated or otherwise available in this sample course of action 908 have been distilled to two parameters 910. In this case, the parameters are IP addresses (e.g., a destination IP address or a source IP address) as noted. Once selected, the source and type of the parameter selected are represented in a shorthand data string in the lookup box 906b as shown in Fig. 9B.

This means the data selection process is much simpler and less error prone. For example, if a task requires an input type of “date”, then only parameters that are of type “date” can show up in the selection panel. As another example, if a task requires an input type of “IP address”, then only parameters that are of type “IP Address” can show up in the selection panel. If the user does not see a particular data element desired for use as an input, this may indicate a need to incorporate a new task before the present task so that the needed

data is available. Such reorganization, modification, and addition of tasks is easy because of the drag and drop functionality of the palate and the parameter definition associated with each task, which is served up to the GUI via a CoA Builder application protocol interface (API), which is used to pull information from the plug-in registration tables up to the CoA builder GUI.

The CoA Builder module can also allow for simple incorporation of additional parameters not previously defined by an upstream plug-in task. A "custom" parameter can be incorporated as indicated in operation 222 in Fig. 2. The option of providing a custom parameter is shown in Fig. 9A as a custom value selection option 912. This value would be hard coded rather than selecting a CoA parameter which would be more contextual to the specific security incident being remediated. As indicated in Fig. 2, once the parameter values have been selected, the configured task is saved in the CoA workflow, e.g., as an XML file, as indicated in operation 224.

Once a task is configured, all output parameter information is referenced as part of the CoA as well, as indicated in operation 226, and is made available to any downstream tasks in the CoA when configuring those tasks. The output parameters are pre-determined based on plug-in command setup. The output parameters are the data available to downstream CoA tasks to incorporate in to either inputs to their commands or condition statements for evaluation and path determination. The result of one task becomes an input to the start of another task.

An integral piece to the CoA Builder module is the normalization and simplification of data within the task, which is one level deeper than the data aliasing described in the previous section. This creates interoperability between different devices by linking the output parameters of one plug-in to input parameters of another. The CoA API normalizes and serves the data up into the XML data layer underlying the user interface to simplify this process. The CoA API is designed to pull information from the plug-in registration tables. When doing so it pulls and maintains in memory all the available output parameters for configured tasks. When configuring a new task, the input parameters for that task as well as their type are also known. The CoA API filters and displays only the output parameters that are of the correct type for input in to the input parameter of the new task.

Fig. 10 is a flow diagram of an exemplary CoA 1000 presented on a workflow palate that may be executed in by the Orchestration Engine to screen a potentially suspicious e-mail based upon the qualities of the proposed recipient. The CoA 1000 begins at a start node 1002 which includes a Start Event Data file 1004 for populating the tasks and conditions in the

CoA 1000. The start event data 1004 may include significantly more data or parameters than is made available to the user when building the CoA 1000 in the GUI. As noted above, the CoA Builder filters the data stored within the CoA 1000 and presents the relevant data usable by a command underlying a task for selection during the construction of the CoA 1000. In this CoA 1000, the first task is to determine the identity of a network user addressee of an e-mail as represented by the determine user task 1008. To perform this function, the determine user task 1008 can have a single input parameter 1006, namely the E-mailAddress of the user. The command underlying the determine user task 1008, via the Orchestration Engine, instantiates the appropriate plug-in (e.g., an interface with the enterprise electronic mail system), which is able to determine and return the recipient user name and other significant data about the user, which is represented by the Output: Determine User data file 1010.

However, the CoA 1000 "knows" that the data needed from the Output: Determine User data file 1010 for potential downstream tasks in the CoA 1000 are output parameters 2012 for UserName and Grade. These two output parameters 1012 are thus the parameters passed through the remaining workflow.

A gateway element 1016 is the next activity in the CoA 1000. The gateway element 1016 is conditioned to determine whether the user recipient of the e-mail message is "greater than grade 18" in a grade structure in the organization. To determine the grade level, the input parameter 1014 provided to the gateway element 1016 is limited to the Grade data output as a parameter from the first task 1008. If the determination is that the user recipient is above grade 18, the path diverts to a quarantine user task 1020. The input parameter 1018 for the quarantine user task 1020 is one of the output parameters 1012 from the determine user task 1008. In this instance, because of the high grade level of the possible user recipient (e.g., a user with a high security clearance), the command underlying the quarantine user task 1020 actuates a plug-in associated with the e-mail server to prevent the potential recipient (the user) from receiving the e-mail. The CoA 1000 then terminates at end node 1022.

Alternatively, if the potential recipient user is below grade level 18, then transmitting an e-mail with a potential threat is determined to be an acceptable risk by the CoA 1000 process. Along this alternate path, an e-mail user task 1026 is provided to pass the e-mail in question to the user. The input parameter 1024 for the e-mail user task 1026 is the E-mailAddress value provided in the Start Event Data 1004 at with the start node 1002. The command underlying the e-mail user task 1026 actuates a plug-in associated with the e-mail server authorizing the e-mail server to send the e-mail to the user. The CoA 1000 then terminates at end node 1028.

The sample data interchange file structure presented below, for example, as a JSON file, is an exemplary representation of a single task. The task representation defines such things as operation sequence, next task, specific plug-in details referenced, and parameter configuration.

```
"deviceTasks" : [{

        "outgoing" :
        ["SequenceFlow_071enfy"], "incoming"
        : ["SequenceFlow_1cdsn3c"],
        "autoOpen" : False,
        "pluginId" : "be29eabb-59fb-4075-b987-13a37d2ede24",
        "deviceId" : "aa1164e6-740b-4cd5-b1cd-f55b3816676e",
        "commandId" : "98a29a08-2e2e-434c-abb5-
        94b659fcccd6",
    "output" : { },
    "input" : {|
        "parameters" : [{
            "value" : {
                "type" : "emailAddress | String",

                "value" : "${COA.emailAddress}"

            },

        }]

"description" :
"DetermineUser" "id" :
"Task_08m97fy"
```

Each task within a CoA would be represented in a similar manner as depicted above.

The several tags and corresponding values in the exemplary task file are explained further as follows:

(1) The "outgoing" string defines the next step in the path of the CoA. If there is more than one path leading outbound from this task, then multiple outgoing values would be defined.

(2) The "incoming" string defines a previous task that can call this task. If there is more than one path leading inbound to this task, then multiple incoming values would be defined.

(3) The "commandId" string identifies the command ID for specific operation (possibly out of several) provided by a plug-in that relates to this task.

(4) The "pluginId" string identifies the ID for the plug-in being used in this task.

(5) The "deviceId" string identifies the specific device instantiation of a plug-in for this task.

(6) The "output" string lists any custom defined output parameters for this task file. Output parameters are otherwise defined in the plug-in command itself and not saved in the JSON.

(7) The "input" string defines the configuration of input parameters for the task commands to use when run.

(8) The "parameter" section stores each of the input parameters required for input to run the task commands.

(9) The "type" string defines the type of data required for input.

(10) The "value" string defines the location of the value to be input, either from the CoA or a custom value manually input.

(11) The "description" and "id" identify the specific task name and identifier within the CoA.

The Orchestration Engine is able to consume the format and recognize exactly what plug-in operations to execute, in what sequence, and where to locate the data need to execute the task.

An exemplary special purpose computer system 1100 for implementing the CoA Builder module and related processes above is depicted in FIG. 11. The computer system 1100 of user of the CoA Builder may be a personal computer (PC), a workstation connected with a server, a notebook or portable computer, a tablet PC, a handheld media player (e.g., an MP3 player), a smart phone device, or other device with internal processing and memory components as well as interface components for connection with external input, output, storage, network, and other types of peripheral devices. Internal components of the computer system in FIG. 11 are shown within the dashed line and external components are shown

outside of the dashed line. Components that may be internal or external are shown straddling the dashed line. Alternatively to a PC, the computer system 1100 instantiating the CoA Builder application may be in the form of any of a server, a mainframe computer, a distributed computer, an Internet appliance, or other computer devices, or combinations thereof.

In any embodiment or component of the system described herein, the computer system 1100 includes a processor 1102 and a system memory 1106 connected by a system bus 1104 that also operatively couples various system components. There may be one or more processors 1102, e.g., a single central processing unit (CPU), or a plurality of processing units, commonly referred to as a parallel processing environment (for example, a dual–core, quad–core, or other multi–core processing device). The system bus 1104 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, a switched-fabric, point-to-point connection, and a local bus using any of a variety of bus architectures. The system memory 1106 includes read only memory (ROM) 1108 and random access memory (RAM) 1110. A basic input/output system (BIOS) 1112, containing the basic routines that help to transfer information between elements within the computer system 1100, such as during start-up, is stored in ROM 1108. A cache 1114 may be set aside in RAM 1110 to provide a high speed memory store for frequently accessed data.

A memory drive interface 1116 may be connected with the system bus 1104 to provide read and write access to a non-volatile data storage device 1118, e.g., a hard disk drive or solid state drive, for nonvolatile storage of applications, files, and data. A number of program modules and other data may be stored on the data storage device 1118, including an operating system 1120, one or more application programs 1122, and data files 1124. In an exemplary implementation, the hard disk drive 1118 may store the course of action builder module 1164, the orchestration engine 1166, the network device plug-ins and related APIs 1168, and the libraries 1170 and related data, e.g., of courses of action, plug-in commands and associated parameters, device registrations, etc. Note that the data storage device 1118 may be either an internal component or an external component of the computer system 1100 as indicated by the data storage device 1118 straddling the dashed line in FIG. 11. In some configurations, there may be both an internal and an external data storage device 1118.

The computer system 1100 may further include an external memory drive 1130 for reading from or writing to a removable magnetic disk 1132, tape, or other magnetic media, or solid state media such as flash memory. Additionally or alternatively, an optical disk drive 1136 for reading from or writing to a removable optical disk 1138 such as a CD ROM or

other optical media may be included in the computer system 1100. The external memory drive 1130 and optical disk drive 1136 may be connected with the system bus 1104 via an external storage interface 1128 to provide read and write access to the external memory drive 1130 or the optical disk drive 1136 initiated by other components or applications within the computer system 1100. The magnetic disk drive 1130 and optical disk drive 1136, and the associated computer-readable media, may be used to provide nonvolatile storage of computer-readable instructions, data structures, program modules, and other data for the computer system 1100.

A display device 1142, e.g., a monitor, a television, or a projector, or other type of presentation device may also be connected to the system bus 1104 via an interface, such as a video adapter 1140 or video card. Similarly, audio devices, for example, external speakers or a microphone (not shown), may be connected to the system bus 1104 through an audio card or other audio interface (not shown).

In addition to the monitor 1142, the computer system 1100 may include other peripheral input and output devices, which are often connected to the processor 1102 and memory 1106 through the serial port interface 1144 that is coupled to the system bus 1106. Input and output devices may also or alternately be connected with the system bus 1104 by other interfaces, for example, a universal serial bus (USB), an IEEE 1394 interface ("Firewire"), a parallel port, or a game port. A user may enter commands and information into the computer system 1100 through various input devices including, for example, a keyboard 1146 and pointing device 1148, for example, a mouse. Other input devices (not shown) may include, for example, a stylus pad, a joystick, a game pad, a tablet, a touch screen device, a satellite dish, a scanner, a facsimile machine, a microphone, a digital camera, and a digital video camera.

Output devices may include a printer 1150 and one or more loudspeakers 1170. Other output devices (not shown) may include, for example, a plotter, a photocopier, a photo printer, a 3-D printer, a facsimile machine, and a press. In some implementations, several of these input and output devices may be combined into single devices, for example, a printer/scanner/fax/photocopier. It should also be appreciated that other types of computer-readable media and associated drives for storing data, for example, magnetic cassettes or flash memory drives, may be accessed by the computer system 1100 via the serial port interface 1144 (e.g., USB) or similar port interface.

The computer system 1100 may operate in a networked environment using logical connections through a network interface 1152 coupled with the system bus 1104 to

communicate with one or more remote devices. The logical connections depicted in FIG. 11 include a local-area network (LAN) 1154 and a wide-area network (WAN) 1160. Such networking environments are commonplace in home networks, office networks, enterprise-wide computer networks, and intranets. These logical connections may be achieved by a communication device coupled to or integral with the computer system 1100. As depicted in FIG. 11, the LAN 1154 may use a router 1156 or hub, either wired or wireless, internal or external, to connect with remote devices, e.g., a remote computer server 1158 or network appliance devices 1157 (e.g., routers, firewalls, etc.) similarly connected on the LAN 1154. For example, in the context of the present application, the Orchestration Engine 1166 operating on the special purpose computer 1100 to execute a course of action can communicate instructions or actions from a plug-in identified in a task over the LAN 1154 to such network appliance devices 1157 to mitigate a cyber security threat on the LAN 1154. The remote computer 1158 connected via the LAN 1154 may be another personal computer, a laptop, a tablet, a smartphone, a server, a client, a peer device, or other common network node, and typically includes many or all of the elements described above relative to the computer system 1100.

To connect with a WAN 1160, the computer system 1100 typically includes a modem 1162 for establishing communications over the WAN 1160. Typically the WAN 1160 may be the Internet. However, in some instances the WAN 1160 may be a large private network spread among multiple locations, or a virtual private network (VPN). The modem 1162 may be a telephone modem, a high speed modem (e.g., a digital subscriber line (DSL) modem), a cable modem, or similar type of communications device. The modem 1162, which may be internal or external, is connected to the system bus 1104 via the network interface 1152. In alternate embodiments the modem 1162 may be connected via the serial port interface 1144. It should be appreciated that the network connections shown are exemplary and other means of and communications devices for establishing a network communications link between the computer system and other devices or networks may be used.

The technology described herein may be implemented as logical operations and/or modules in one or more systems. The logical operations may be implemented as a sequence of processor-implemented steps executing in one or more computer systems and as interconnected machine or circuit modules within one or more computer systems. Likewise, the descriptions of various component modules may be provided in terms of operations executed or effected by the modules. The resulting implementation is a matter of choice, dependent on the performance requirements of the underlying system implementing the

described technology. Accordingly, the logical operations making up the embodiments of the technology described herein are referred to variously as operations, steps, objects, or modules. Furthermore, it should be understood that logical operations may be performed in any order, unless explicitly claimed otherwise or a specific order is inherently necessitated by the claim language.

In some implementations, articles of manufacture are provided as computer program products that cause the instantiation of operations on a computer system to implement the procedural operations. One implementation of a computer program product provides a non-transitory computer program storage medium readable by a computer system and encoding a computer program. It should further be understood that the described technology may be employed in special purpose devices independent of a personal computer.

The above specification, examples and data provide a complete description of the structure and use of exemplary embodiments of the invention as defined in the claims. Although various embodiments of the claimed invention have been described above with a certain degree of particularity, or with reference to one or more individual embodiments, those skilled in the art could make numerous alterations to the disclosed embodiments without departing from the spirit or scope of the claimed invention. Other embodiments are therefore contemplated. It is intended that all matter contained in the above description and shown in the accompanying drawings shall be interpreted as illustrative only of particular embodiments and not limiting. Changes in detail or structure may be made without departing from the basic elements of the invention as defined in the following claims.

Some embodiments described herein relate to a computer storage product with a non-transitory computer-readable medium (also can be referred to as a non-transitory processor-readable medium) having instructions or computer code thereon for performing various computer-implemented operations. The computer-readable medium (or processor-readable medium) is non-transitory in the sense that it does not include transitory propagating signals per se (e.g., a propagating electromagnetic wave carrying information on a transmission medium such as space or a cable). The media and computer code (also can be referred to as code) may be those designed and constructed for the specific purpose or purposes. Examples of non-transitory computer-readable media include, but are not limited to: magnetic storage media such as hard disks, floppy disks, and magnetic tape; optical storage media such as Compact Disc/Digital Video Discs (CD/DVDs), Compact Disc-Read Only Memories (CD-ROMs), and holographic devices; magneto-optical storage media such as optical disks; carrier wave signal processing modules; and hardware devices that are specially configured to

store and execute program code, such as Application-Specific Integrated Circuits (ASICs), Programmable Logic Devices (PLDs), Read-Only Memory (ROM) and Random-Access Memory (RAM) devices. Other embodiments described herein relate to a computer program product, which can include, for example, the instructions and/or computer code discussed herein.

Examples of computer code include, but are not limited to, micro-code or microinstructions, machine instructions, such as produced by a compiler, code used to produce a web service, and files containing higher-level instructions that are executed by a computer using an interpreter. For example, embodiments may be implemented using imperative programming languages (e.g., C, Fortran, etc.), functional programming languages (Haskell, Erlang, etc.), logical programming languages (e.g., Prolog), object-oriented programming languages (e.g., Java, C++, etc.) or other suitable programming languages and/or development tools. Additional examples of computer code include, but are not limited to, control signals, encrypted code, and compressed code.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. Where methods described above indicate certain events occurring in certain order, the ordering of certain events may be modified. Additionally, certain of the events may be performed concurrently in a parallel process when possible, as well as performed sequentially as described above.

What is claimed is:

1. A method of facilitating user interactions with a graphical user interface to build a workflow for responding to a security event on a computing network, the graphical user interface generated and rendered on a display of a computer system connected to the computing network, by executing software with a processor of the computer system, the method comprising:

providing a plurality of task stencils within the graphical user interface for selection by a user to design the workflow, each task stencil from the plurality of task stencils representing a task to be performed by a plurality of hardware devices to respond to the security event;

receiving from the user a selection of a task stencil from the plurality of task stencils, the task stencil corresponding to one or more tasks from a plurality of tasks;

receiving from the user a selection of a plug-in module from a plurality of plug-in modules, each plug-in module of the plurality of plug-in modules having one or more activation commands for activating a corresponding hardware device of the plurality of hardware devices connected to the computing network to perform a task from the plurality of tasks;

providing a first configuration interface to associate an activation command of the plug-in module with the one or more tasks;

providing a second configuration interface to associate input data, output data, and parameter configuration data for use in executing the activation command with the one or more tasks.

2. The method of claim 1, further comprising:

responding to the security event by executing the activation command to activate, over the computing network, the corresponding hardware device to protect the computing network.

3. The method of claim 1, further comprising:

responding to the security event by executing the activation command to activate, over the computing network, the corresponding hardware device to protect the computing network, executing the activation command including configuring the corresponding hardware device to block or remediate the security event.

4. The method of claim 1, further comprising:

receiving, by the computer system, a message to trigger execution of the activation command;

responding to the security event by executing the activation command to activate, over the computing network, the corresponding hardware device to protect the computing network,

5. The method of claim 1, wherein the graphical user interface includes the first configuration interface and the second configuration interface, the method further comprising:

representing the plurality of task stencils on the display as representational elements having “drag and drop” features.

6. The method of claim 1, further comprising:

building a data interchange file associated with the one or more tasks associated with the activation command, input data, output data, and parameter configuration data for use in executing the activation command.

7. A method of facilitating user interactions with a graphical user interface to build a workflow for responding to a security event on a computing network, the graphical user interface generated and rendered on a display of a computer system connected to the computing network, by executing software with a processor of the computer system, the method comprising:

using the graphical user interface to present to a user a plurality of task stencils available for selection by the user to design a workflow, each task stencil from the plurality of task stencils representing at least one task;

enabling, with the graphical user interface, the user to select (a) a start node stencil graphically representing an adapter; and b) one or more task stencils from the plurality of task stencils;

enabling, with the graphical user interface, the user to select a graphical representation of a plug-in module from a plurality of graphical representations of plug-in modules, each graphical representation of a plug-in module representing a corresponding network device on the computing network and the plug-in module for that graphical representation providing an

activation command for activating the corresponding network device, so that, when executed, the activation command causes the corresponding network device to perform an action in response to the security event;

enabling, with the graphical user interface, the user to associate at least one of the activation commands of each of the selected graphical representations of plug-in modules with the one or more tasks corresponding to the selected task stencils;

enabling, with the graphical user interface, the user to arrange the workflow to establish a sequence of tasks including the one or more tasks corresponding to the selected task stencils and conditions by which each task of the sequence of tasks is operated.

8. The method of claim 7, wherein the adapter corresponding to the start node stencil includes parameters used in executing the activation command.

9. The method of claim 7, wherein the graphical user interface references a device table containing information regarding the corresponding network devices on the computing network in enabling the user to select the graphical representations of plug-in modules.

10. The method of claim 7, further comprising:

registering the corresponding network device by associating a device identification parameter of the corresponding network device with the selected graphical representation of the plug-in module.

11. The method of claim 7, wherein the activation commands are aliased to common and human understandable actions.

12. The method of claim 7, wherein each activation command includes at least one of block Internet Protocol (IP) address, or an instruction to activate a firewall.

13. The method of claim 7, further comprising:

enabling, with the graphical user interface, the user to select an end node stencil graphically representing an end node, the end node representing an end of the workflow.

14. The method of claim 7, wherein:

the one or more selected task stencils include a first task stencil and a second task stencil, the one or more graphical representations of plug-in modules includes a first graphical representation of plug-in module and a second graphical representation of plug-in module,

the sequence of tasks including the first task stencil and the second task stencil.

15. The method of claim 7, further comprising:

receiving, at the graphical user interface, input data associated with the one or more tasks associated with the selected task stencils, the input data being associated with output data from a task prior to the one or more tasks in the workflow.

16. The method of claim 7, wherein selecting the one or more task stencils from the plurality of task stencils includes dragging and dropping the one or more task stencils.

17. The method of claim 7, wherein selecting the one or more graphical representations of plug-in modules includes dragging and dropping the one more graphical representations of plug-in modules.

18. The method of claim 7, further comprising:

sending the at least one of the activation commands to each corresponding network device from a plurality of network device to execute the one or more tasks corresponding to the selected one or more task stencils.

19. The method of claim 7, further comprising:

enabling the graphical user interface to receive input data, output data, and parameter configuration data for use in executing the activation command associated with the one or more tasks corresponding to the selected one or more task stencils.

20. A non-transitory storage medium that includes a computer program executable by a processor of a computer system, the computer program comprising:

computer-readable instructions to provide a plurality of task stencils within a graphical user interface for selection by a user to design a workflow, each task stencil from

the plurality of task stencils representing a task to be performed by a plurality of hardware devices to respond to a security event;

computer-readable instructions to receive from the user a selection of a task stencil from the plurality of task stencils, the task stencil associated with one or more tasks from a plurality of tasks;

computer-readable instructions to receive from the user a selection of a plug-in module from a plurality of plug-in modules, each plug-in module of the plurality of plug-in modules having one or more activation commands for activating a corresponding hardware device of the plurality of hardware devices connected to the computing network to perform a task from the plurality of tasks;

computer-readable instructions to provide a first configuration interface to associate an activation command of the plug-in module with the one or more tasks;

computer-readable instructions to provide a second configuration interface to associate input data, output data, and parameter configuration data for use in executing the activation command with the one or more tasks.

21. A method, comprising:

receiving, at a graphical user interface implemented by a processor, from a user a selection of a first task stencil from a plurality of task stencils and a second task stencil from the plurality of task stencils, each task stencil from the plurality of task stencils representing a task from a plurality of tasks, the plurality of tasks corresponding to responses executable by a plurality of network devices to a security event, the plurality of network devices operatively coupled to the processor via a computer network;

receiving, at the graphical user interface, from the user (1) a selection of a first plug-in module from a plurality of plug-in modules, the first plug-in module being associated with the first task, and (2) a selection of a second plug-in module from the plurality of plug-in modules, the second plug-in module being associated with the second task, each plug-in module from the plurality of plug-in modules corresponding to each network device from the plurality of network devices;

associating a first activation command of the first plug-in module with the first task and a second activation command of the second plug-in module with the second task;

generating a workflow including the first activation command associated with a first network device from the plurality of network devices and the second activation command associated with a second network device from the plurality of network devices and in an

order relative to the first activation command, input data associated with the second activation command associated with output data from the first activation command; and

sending the first activation command to the first network device for execution in response to the security event and the second activation command to the second network device for execution in response to the security event.

22. A system, comprising:

a memory; and

a processor operatively coupled to the memory, the processor configured to:

receive, at a graphical user interface, from a user a selection of a first task stencil from a plurality of task stencils and a second task stencil from the plurality of task stencils, each task stencil from the plurality of task stencils representing a task from a plurality of tasks, the plurality of tasks corresponding to responses executable by a plurality of network devices to a security event, the plurality of network devices operatively coupled to the processor via a computer network;

receive, at the graphical user interface, from the user (1) a selection of a first plug-in module from a plurality of plug-in modules, the first plug-in module being associated with the first task, and (2) a selection of a second plug-in module from the plurality of plug-in modules, the second plug-in module being associated with the second task, each plug-in module from the plurality of plug-in modules corresponding to each network device from the plurality of network devices;

associate a first activation command of the first plug-in module with the first task and a second activation command of the second plug-in module with the second task;

generate a workflow including the first activation command associated with a first network device from the plurality of network devices and the second activation command associated with a second network device from the plurality of network devices and in an order relative to the first activation command;

send the first activation command to the first network device for execution in response to the security event and the second activation command to the second network device for execution in response to the security event.

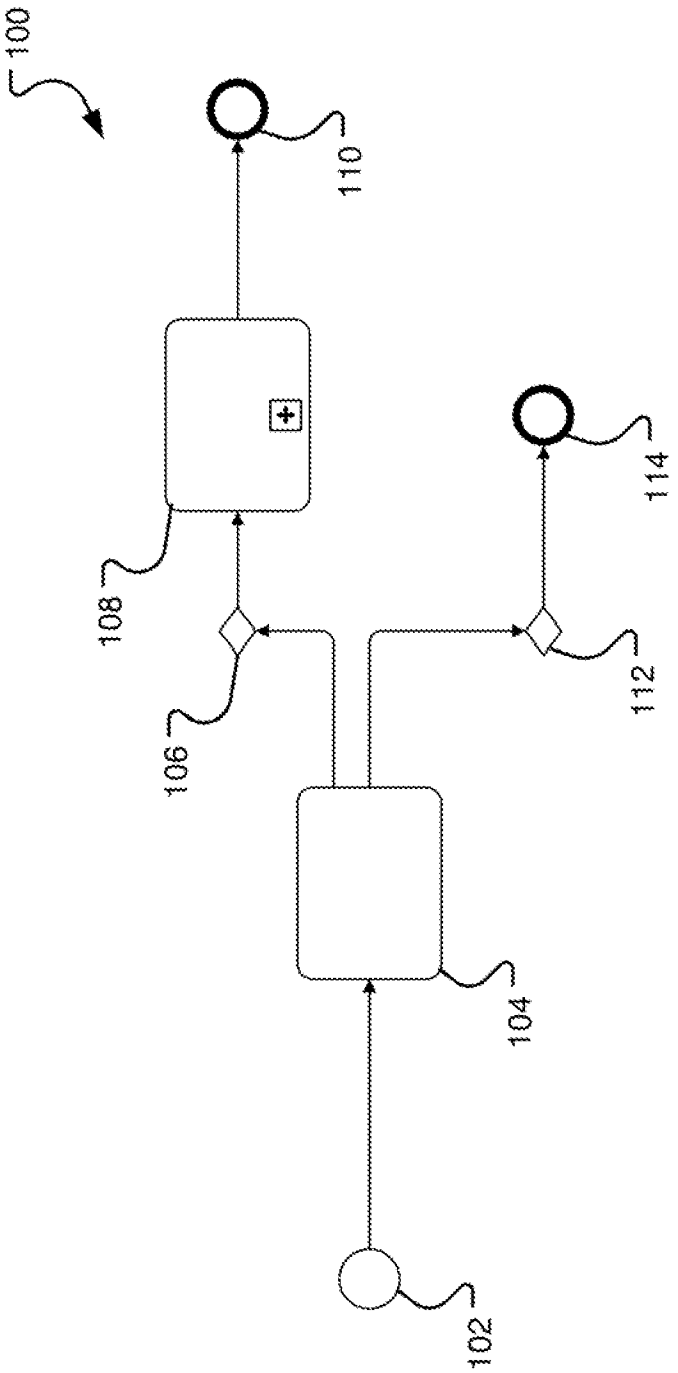


Fig. 1

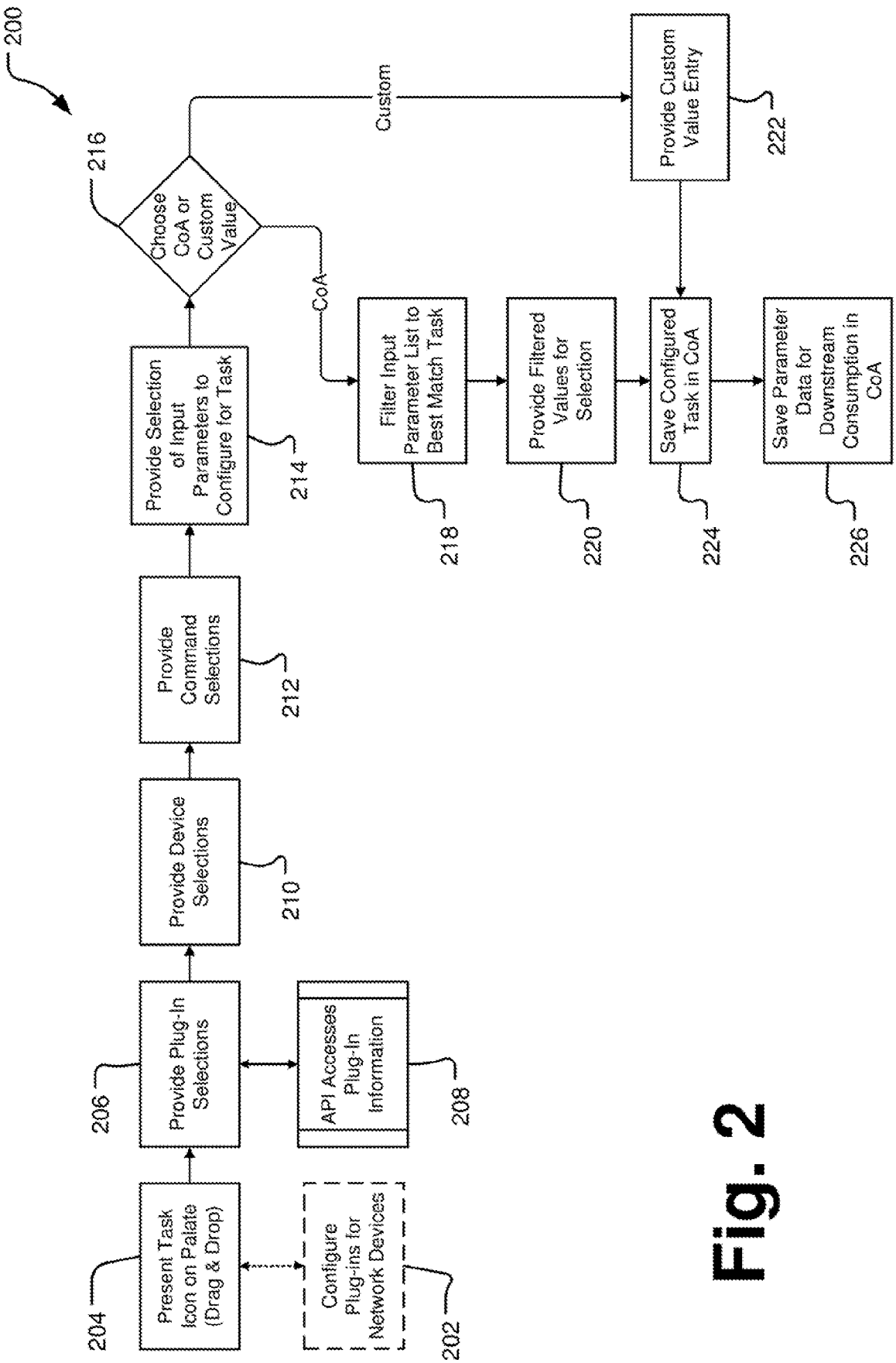

200
216
Choose CoA or Custom Value
Custom
Provide Custom Value Entry
222
CoA
Provide Selection of Input Parameters to Configure for Task
214
Filter Input Parameter List to Best Match Task
218
Provide Filtered Values for Selection
220
Save Configured Task in CoA
224
Save Parameter Data for Downstream Consumption in CoA
226
Provide Command Selections
212
Provide Device Selections
210
Provide Plug-In Selections
206
API Accesses Plug-In Information
208
Present Task Icon on Palate (Drag & Drop)
204
Configure Plug-ins for Network Devices
202

Fig. 2

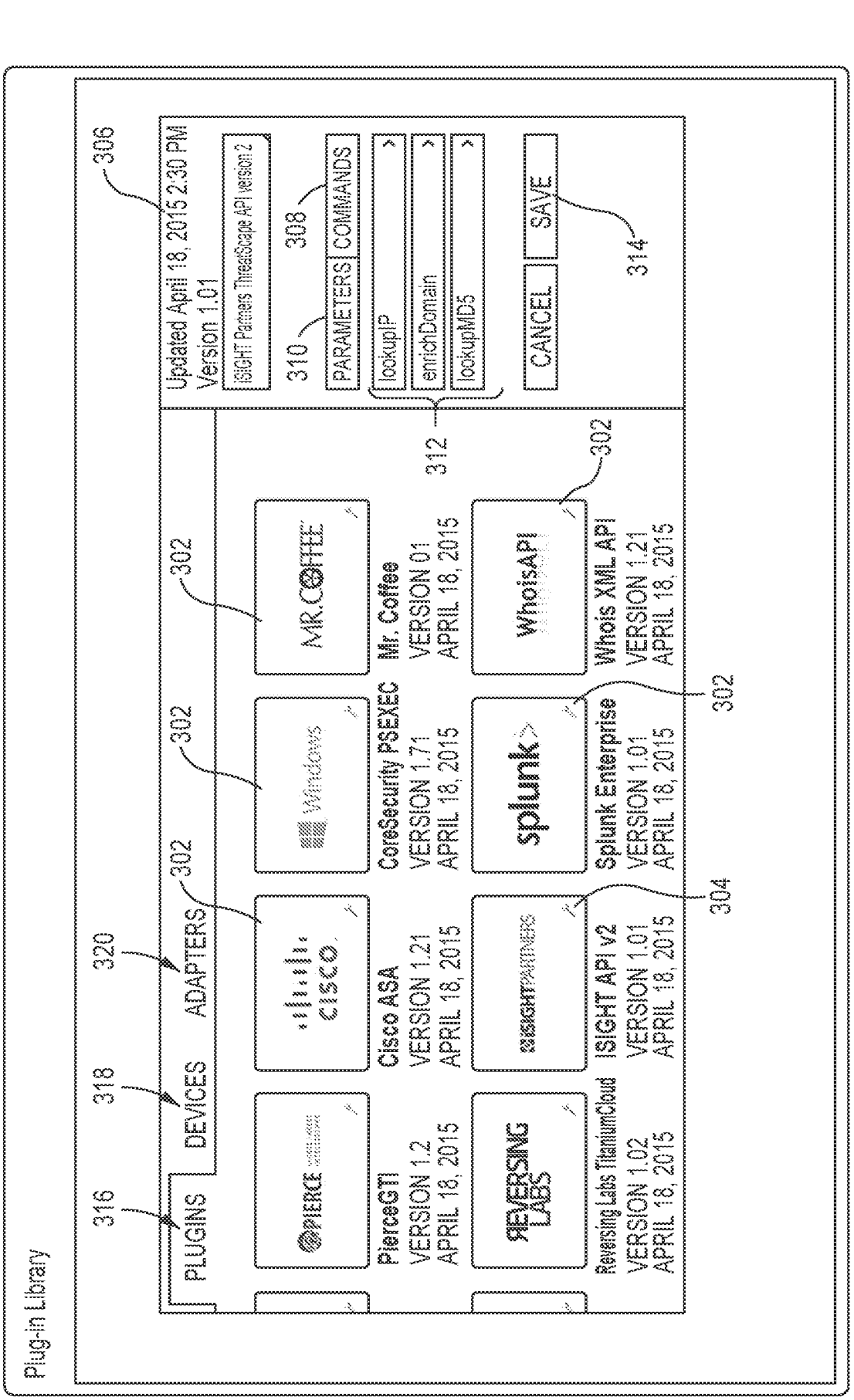

300
Plug-in Library
316
318
320
PLUGINS
DEVICES
ADAPTERS
306
Updated April 18, 2015 2:30 PM
Version 1.01
iSIGHT Partners ThreatScape API version 2
308
310
PARAMETERS
COMMANDS
lookupIP
enrichDomain
lookupMD5
312
CANCEL
SAVE
314
302
PierceGTI
VERSION 1.2
APRIL 18, 2015
Cisco ASA
VERSION 1.21
APRIL 18, 2015
CoreSecurity PSEXEC
VERSION 1.71
APRIL 18, 2015
Mr. Coffee
VERSION 01
APRIL 18, 2015
Reversing Labs TitaniumCloud
VERSION 1.02
APRIL 18, 2015
ISIGHT API v2
VERSION 1.01
APRIL 18, 2015
304
Splunk Enterprise
VERSION 1.01
APRIL 18, 2015
Whois XML API
VERSION 1.21
APRIL 18, 2015

Fig. 3

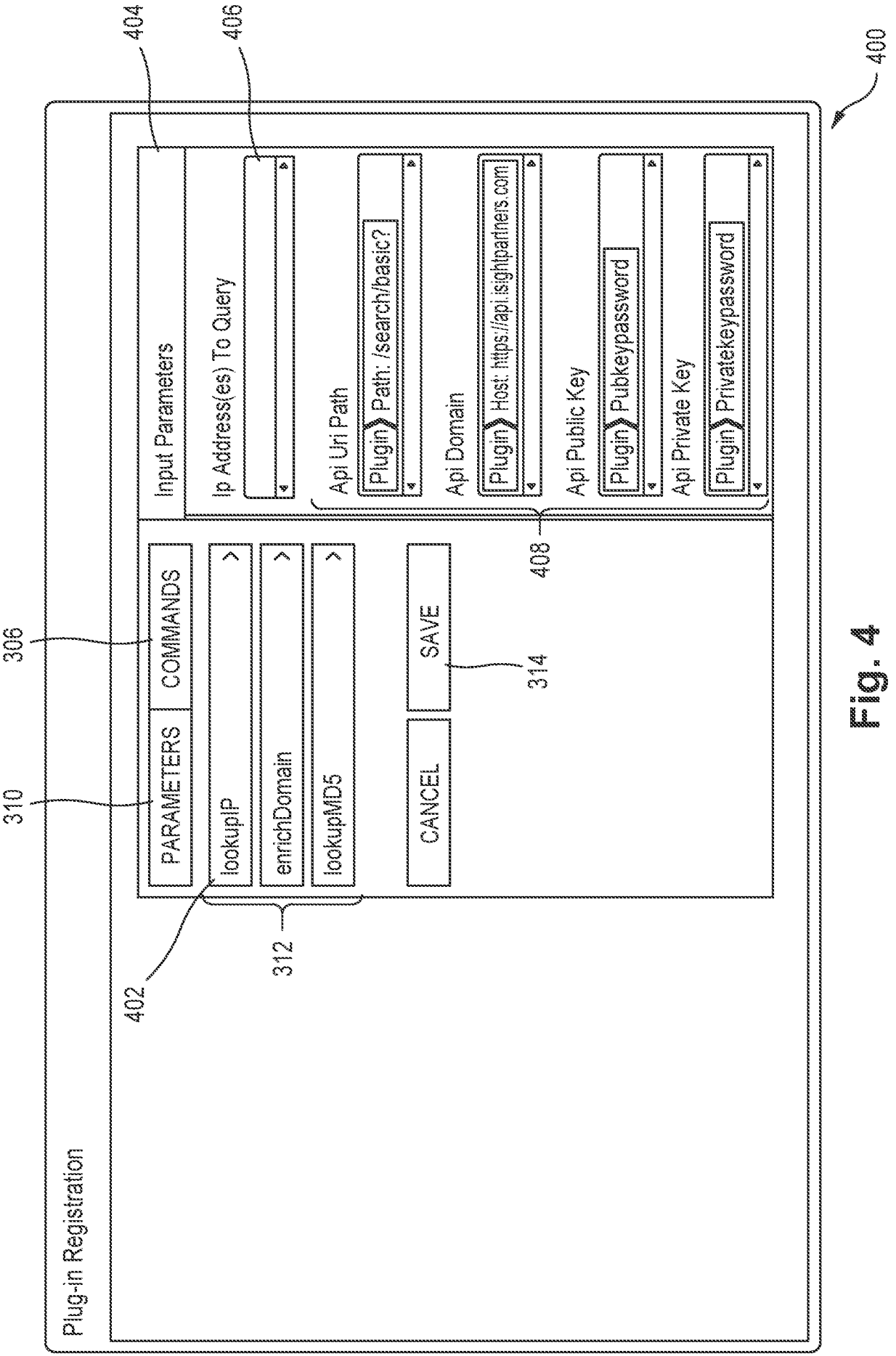

400
404
406
Plug-in Registration
Input Parameters
Ip Address(es) To Query
Api Uri Path
Plugin
Path: /search/basic?
Api Domain
Plugin
Host: https://api.isightpartners.com
Api Public Key
Plugin
Pubkeypassword
Api Private Key
Plugin
Privatekeypassword
408
306
COMMANDS
PARAMETERS
310
lookupIP
enrichDomain
lookupMD5
312
CANCEL
SAVE
314
402

Fig. 4

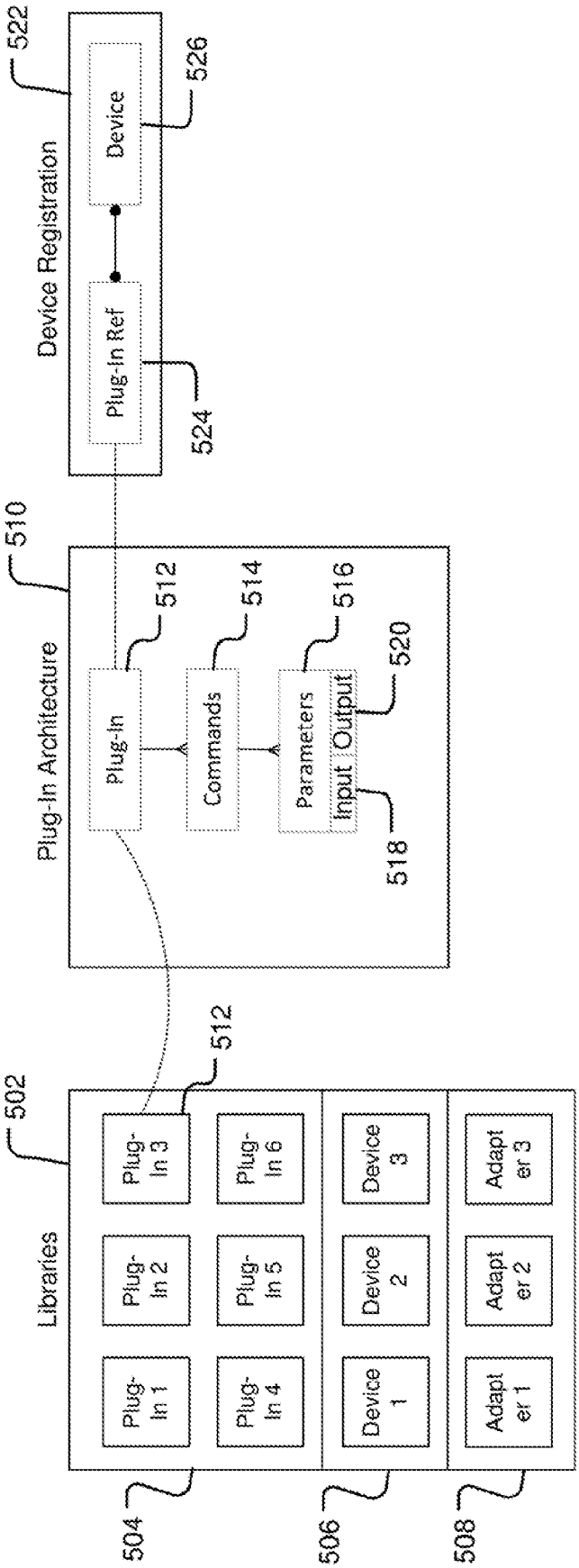

522
Device Registration
Device
526
Plug-In Ref
524
510
Plug-In Architecture
512
Plug-In
514
Commands
516
Parameters
520
Input
Output
518
512
502
Libraries
Plug-In 3
Plug-In 6
Plug-In 2
Plug-In 5
Plug-In 1
Plug-In 4
Device 3
Device 2
Device 1
Adapter 3
Adapter 2
Adapter 1
504
506
508

Fig. 5

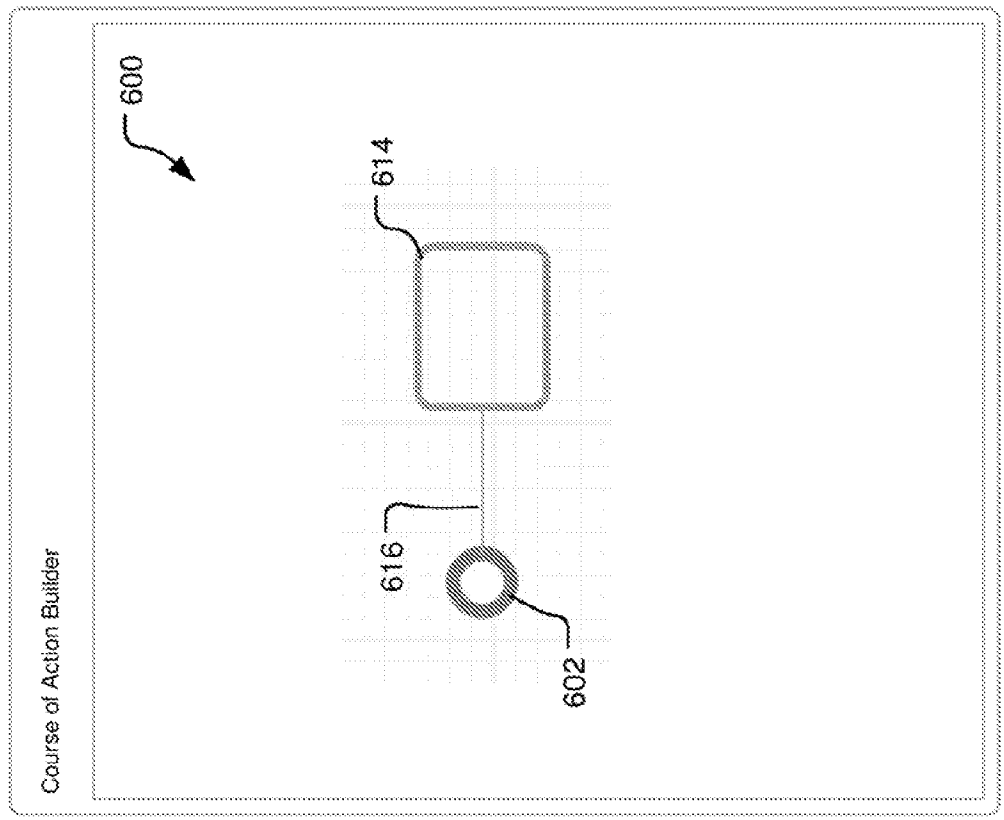

Course of Action Builder
600
614
616
602

Fig. 6B

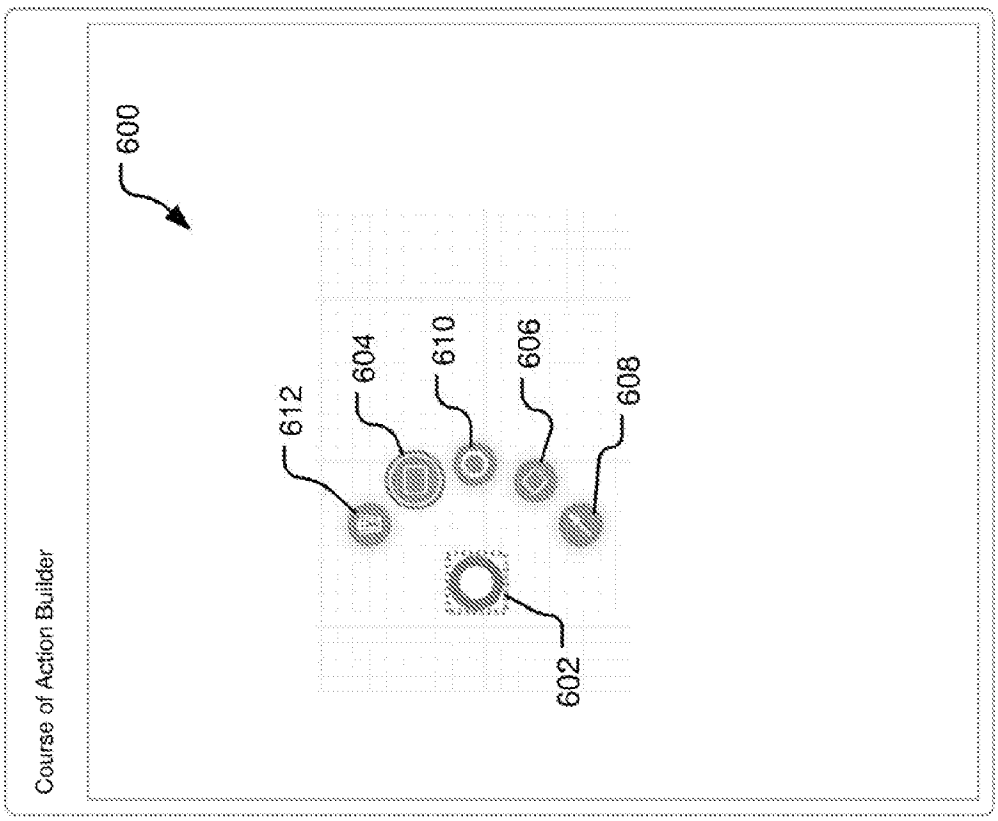

Course of Action Builder
600
612
604
610
606
608
602

Fig. 6A

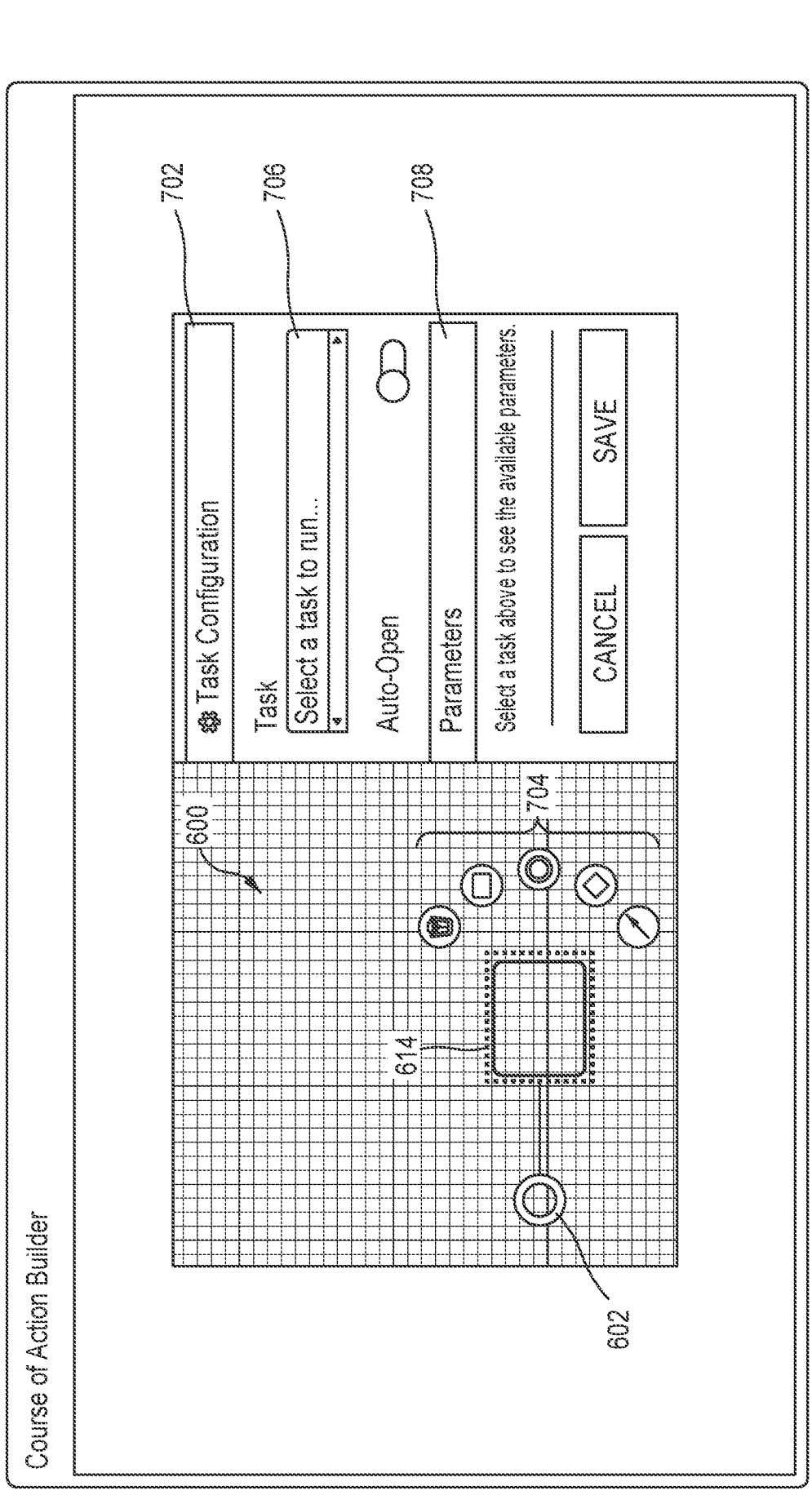

700
Course of Action Builder
600
602
614
702
704
706
708
Task Configuration
Task
Select a task to run...
Auto-Open
Parameters
Select a task above to see the available parameters.
CANCEL
SAVE

Fig. 7

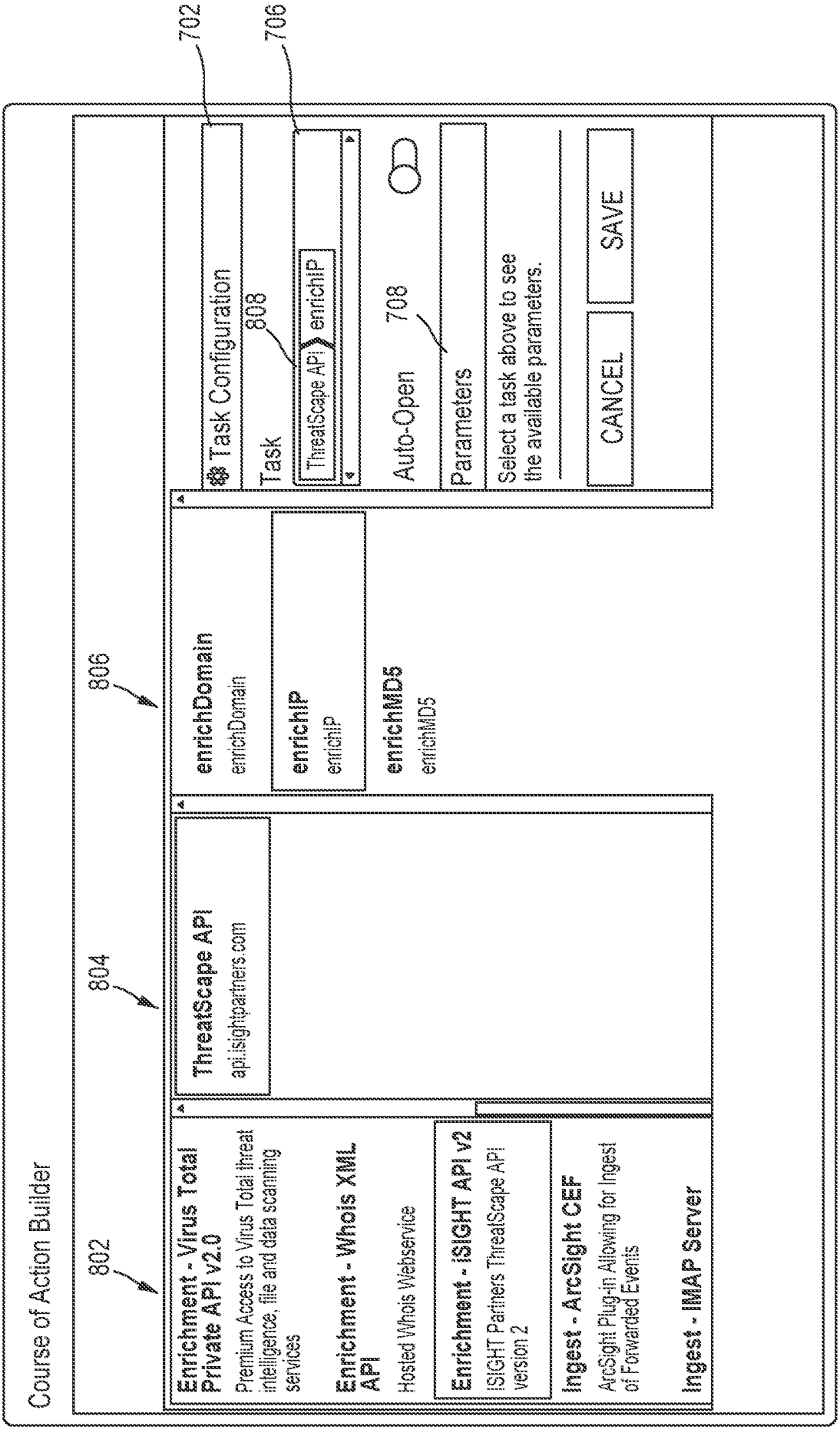

Course of Action Builder
802
804
806
Enrichment - Virus Total Private API v2.0
Premium Access to Virus Total threat intelligence, file and data scanning services
Enrichment - Whois XML API
Hosted Whois Webservice
Enrichment - iSIGHT API v2
iSIGHT Partners ThreatScape API version 2
Ingest - ArcSight CEF
ArcSight Plug-in Allowing for Ingest of Forwarded Events
Ingest - IMAP Server
ThreatScape API
api.isightpartners.com
enrichDomain
enrichDomain
enrichIP
enrichIP
enrichMD5
enrichMD5
Task Configuration
Task
ThreatScape API
enrichIP
Auto-Open
Parameters
Select a task above to see the available parameters.
CANCEL
SAVE
702
706
808
708

Fig. 8

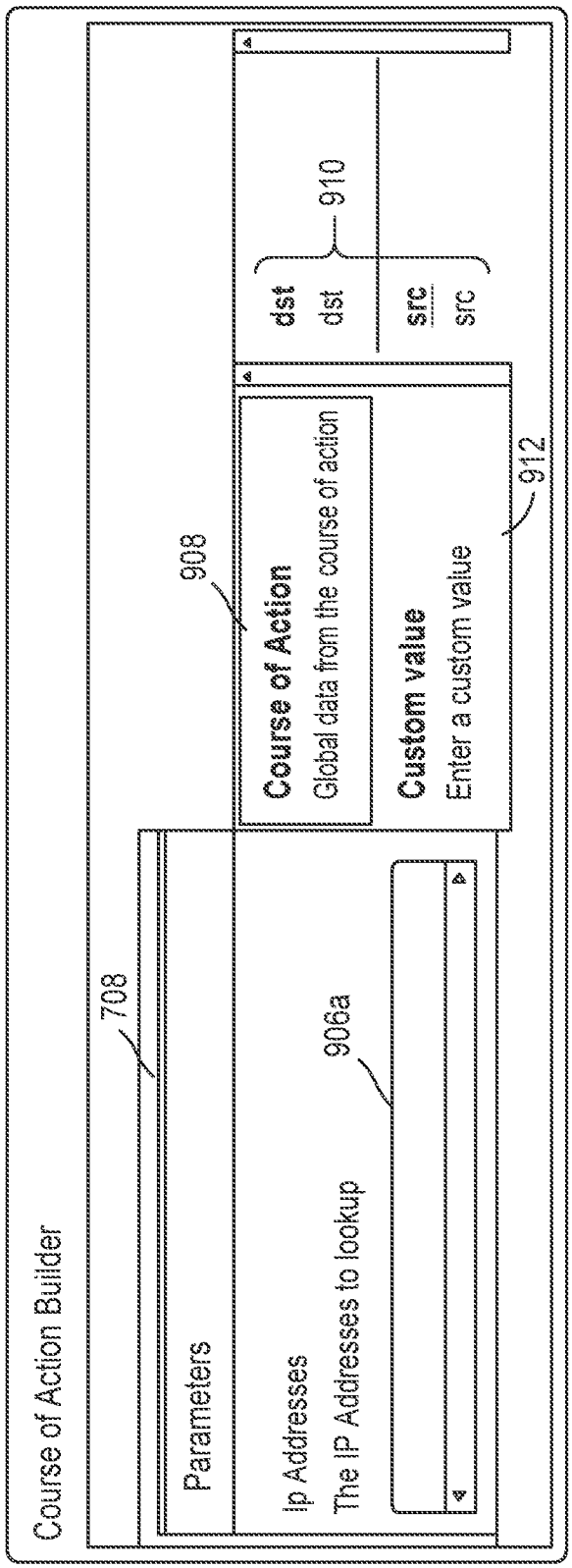

Course of Action Builder
708
Parameters
Ip Addresses
The IP Addresses to lookup
906a
908
Course of Action
Global data from the course of action
Custom value
Enter a custom value
912
dst
dst
src
src
910

Fig. 9A

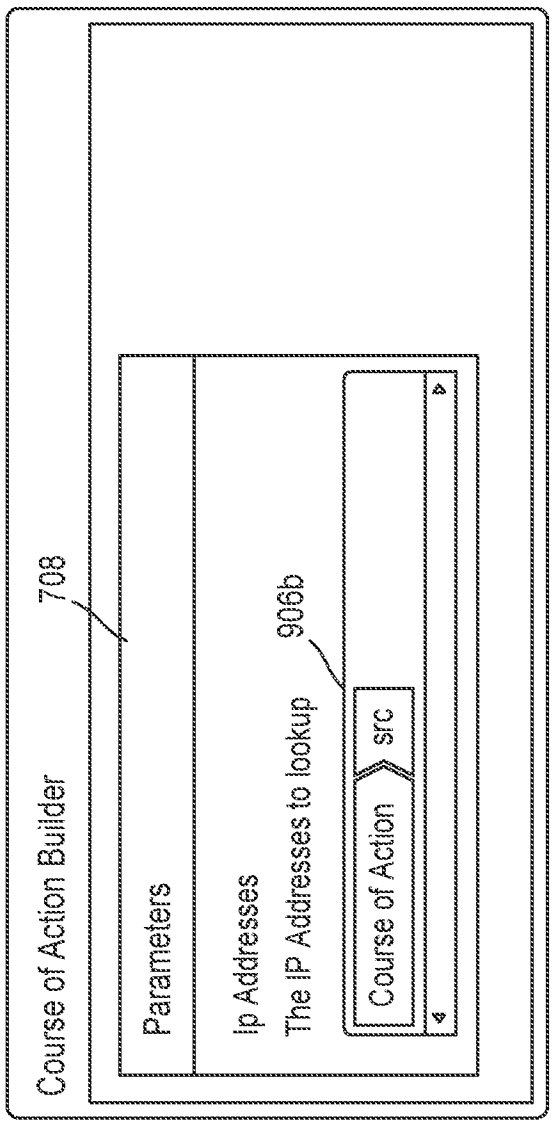

Course of Action Builder
708
Parameters
Ip Addresses
The IP Addresses to lookup
906b
Course of Action
src

Fig. 9B

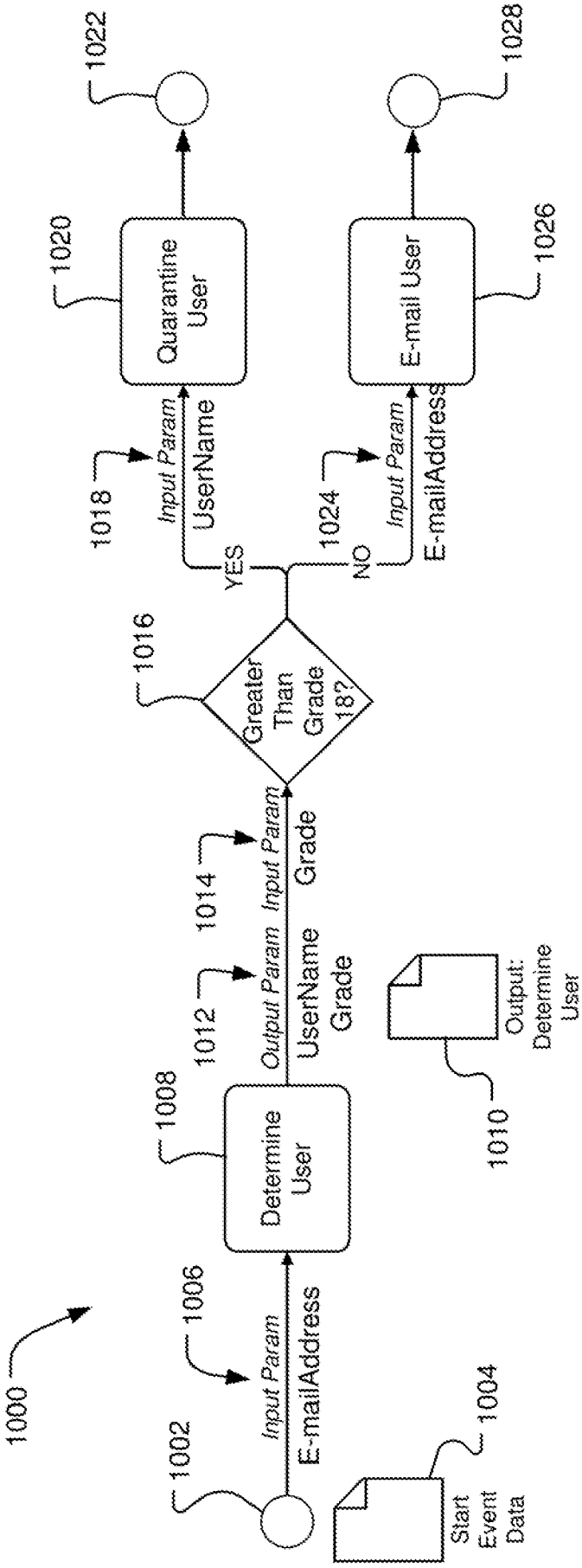
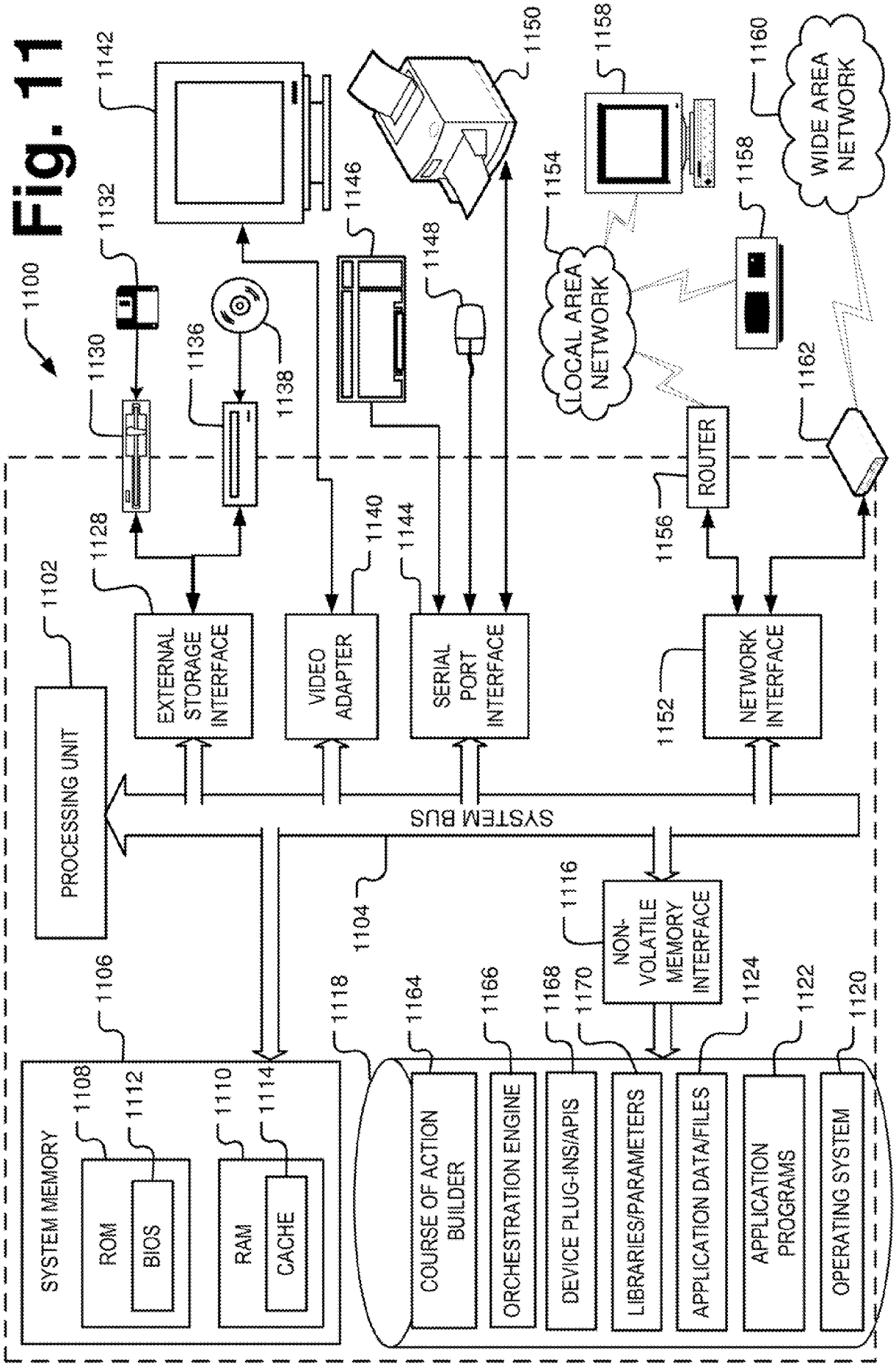

1000
1002
1004
Start Event Data
1006
Input Param
E-mailAddress
1008
Determine User
1010
Output: Determine User
1012
Output Param
UserName
Grade
1014
Input Param
Grade
1016
Greater Than Grade 18?
YES
NO
1018
Input Param
UserName
1020
Quarantine User
1022
1024
Input Param
E-mailAddress
1026
E-mail User
1028

Fig. 10

Fig. 11


1100
1102
PROCESSING UNIT
1104
SYSTEM BUS
1106
SYSTEM MEMORY
1108
ROM
1112
BIOS
1110
RAM
1114
CACHE
1116
NON-VOLATILE MEMORY INTERFACE
1118
1120
OPERATING SYSTEM
1122
APPLICATION PROGRAMS
1124
APPLICATION DATA/FILES
1128
EXTERNAL STORAGE INTERFACE
1130
1132
1136
1138
1140
VIDEO ADAPTER
1142
1144
SERIAL PORT INTERFACE
1146
1148
1150
1152
NETWORK INTERFACE
1154
LOCAL AREA NETWORK
1156
ROUTER
1158
1160
WIDE AREA NETWORK
1162
1164
COURSE OF ACTION BUILDER
1166
ORCHESTRATION ENGINE
1168
DEVICE PLUG-INS/APIS
1170
LIBRARIES/PARAMETERS

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/38809

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 17/00 (2016.01)
CPC - H04L63/20
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
CPC: H04L63/20; IPC(8): G06F 17/00 (2016.01); USPC: 726/1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
CPC: H04L63/20, H04L63/102, G06F21/6218, G06F21/604, H04L63/0227; USPC: 726/1, 726/14, 715/763, 715/762, 715/765 (keyword limited; terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase; Google Scholar
Search Terms: task stencil user design workflow graphical interface hardware security event plug-in activation

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 2008/0040191 A1 (CHAKRAVARTY et al.) 14 February 2008 (14.02.2008), entire document, especially abstract and para [0005], [0010]-[0014], [0025]-[0028], [0030], [0036], [0041]-[0043], [0046], [0048]-[0050], [0052], [0058]-[0062], Fig. 2, Fig. 3.	1-5, 20 ---------- 6
Y	US 2014/0082749 A1 (HOLLAND et al.) 20 March 2014 (20.03.2014), entire document, especially abstract and para [0054]-[0057], [0096], [0113].	6
A	US 2012/0210265 A1 (DELIA et al.) 16 August 2012 (16.08.2012), entire document.	1-6, 20
A	US 2011/0039237 A1 (SKARE) 17 February 2011 (17.02.2011), entire document.	1-6, 20

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

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 October 2016 (17.10.2016)	28 OCT 2016
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/38809

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

-- see extra sheet --

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:
1-6, 20

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 16/38809

In continuation of Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet):

This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1.

Group I: Claims 1-6, 20, directed to a method method of facilitating user interactions with a graphical user interface to build a workflow for responding to a security event on a computing network, further comprising task stencils and plug-in modules having one or more activation commands for activating a corresponding hardware device.

Group II: Claims 7-19, directed to a method of facilitating user interactions with a graphical user interface to build a workflow for responding to a security event on a computing network, further comprising a start node stencil graphically representing an adapter, and graphical representations of plug-in modules.

Group III: Claim 21-22, directed to a method for receiving, at a graphical user interface implemented by a processor, from a user a selection of two task stencils, two plug-in modules, and associating activation commands for each plug-in module with a respective first and second task.

The groups of inventions listed above do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons:

The special technical feature of the Group I invention is each task stencil from the plurality of task stencils representing a task to be performed by a plurality of hardware devices to respond to the security event; each plug-in module of the plurality of plug-in modules having one or more activation commands for activating a corresponding hardware device of the plurality of hardware devices; providing a second configuration interface to associate input data, output data, and parameter configuration data for use in executing the activation command with the one or more tasks, not required by Groups II-III.

The special technical feature of the Group II invention is a start node stencil graphically representing an adapter; a plurality of graphical representations of plug-in modules, each graphical representation of a plug-in module representing a corresponding network device; enabling, with the graphical user interface, the user to arrange the workflow to establish a sequence of tasks including the one or more tasks corresponding to the selected task stencils and conditions by which each task of the sequence of tasks is operated, not required by Groups I or III.

The special technical feature of the Group III invention is a selection of a second plug-in module from the plurality of plug-in modules, the second plug-in module being associated with the second task; associating a first activation command of the first plug-in module with the first task and a second activation command of the second plug-in module with the second task; generating a workflow including the first activation command associated with a first network device from the plurality of network devices and the second activation command associated with a second network device from the plurality of network devices and in an order relative to the first activation command, input data associated with the second activation command associated with output data from the first activation command, not required by Groups I-II.

Groups I-III share the technical features of, in various combination, providing a plurality of task stencils within a graphical user interface for selection by a user to design a workflow, each task stencil from the plurality of task stencils representing at least one task, receiving from a user a selection of a task stencil from the plurality of task stencils, receiving from a user a selection of a plug-in module from a plurality of plug-in modules, one or more activation commands for activating a corresponding device, of a plurality of devices connected to a computer network, associating an activation command of a plug-in module with a task, and a task/action to be performed/executed in response to a security event.

(See Next Extra Sheet)

In continuation of previous Extra Sheet:

However, these shared technical features fail to represent a contribution over the prior art of US 2008/0040191 A1 to Chakravarty et al. (hereinafter 'Chakravarty'), which discloses providing a plurality of task stencils within a graphical user interface for selection by a user to design a workflow (para [0005], [0011]-[0012], [0046], [0048] - "workflow module may be used to create, edit and run workflow instances. The workflow module includes, at least, an edit mode for creating and/or editing workflow instances and a remediation mode for remediating incident objects. The graphical user interface can be used to display information from the incident tracking module and/or workflow module"; "Templates can be created to facilitate creation and editing of workflow elements"), each task stencil from the plurality of task stencils representing at least one task (para [0008], [0011]-[0012], [0041], Fig. 3 - "user interface may include several panels within a window that display various create/edit functionalities. FIG. 3 illustrates a window having a panel for the workflow process instance 202, process elements 203, element properties 204, message panel 205, work-item palette 206, and overview panel 207. The user interface may be customized to include additional panels, alternative panels, toolbars and menu bars in different arrangements"), receiving from a user a selection of a task stencil from the plurality of task stencils (para [0011]-[0012], [0042] - "A user may create and edit a workflow process from scratch or by starting with a template. For example, the use may create or edit a workflow instance by using drag and drop features of the graphical user interface to place one or more steps/work-items and transitions there between into a sequence"; "The panel for workflow process instance 202 enables the user to create/edit a workflow process instance using drag and drop elements from the work-item palette 206 in order to build a graph 201 that represents a desired workflow process"), receiving from a user a selection of a plug-in module from a plurality of plug-in modules (para [0010], [0012]-[0014], [0046], [0049] - "the user interface may be used to create a new workflow process instance and/or edit existing workflow process instances (e.g., by using graphical objects and/or other object oriented tools)"; "For a selected incident object, the user interface may graphically display the selected workflow process instance (e.g., including the steps, with their associated work-item(s), and transitions in the selected workflow process instance between steps)"; "Work-item palette panel 206 lists all the work-item types supported by the system including custom created work-items. The list of work-items may be displayed in icon format or text format. Either format allows the user to drag and drop a selected work-item from work-item palette panel 206 into process panel"), one or more activation commands for activating (para [0050] - "command work-item"; "work-items (template work-item, command work-item, alert work-item, and decision work-item) may be automatically executed to perform a specified action") a corresponding device, of a plurality of devices connected to a computer network (para [0027], [0058] - "Activity monitor 12 may collect events in real-time from security devices, network devices, security software and other devices and applications running on a network"), associating an activation command of a plug-in module with a task (para [0014], [0052] - "A command work-item may be added to execute a configured command that allows parameters to be passed into the command to be executed. The user may specify the command in a text box 502. The command name may be specified in a separate text box 501. The command may be used to execute an action in an attempt to resolve at least a portion of the incident"), and a task/action to be performed/executed in response to a security event (para [0009], [0028], [0052] - "In remediation mode, the workflow module executes the assigned workflow process instance(s) in an attempt to remediate the designated incident. Execution of the workflow process instance may include implementing the steps of the workflow (including one or more work-items in each step)").

Groups II-III share the technical features of, in various combination, providing a plurality of task stencils within a graphical user interface for selection by a user to design a workflow, each task stencil from the plurality of task stencils representing at least one task, receiving from a user a selection of a task stencil from the plurality of task stencils, receiving from a user a selection of a plug-in module from a plurality of plug-in modules, each plug-in module representing a corresponding network device on a computing network, one or more activation commands for activating a corresponding network device, of a plurality of network devices connected to a computer network, associating an activation command of a plug-in module with a task, and a task/action to be performed/executed in response to a security event.

However, these shared technical features fail to represent a contribution over the prior art of Chakravarty, which discloses providing a plurality of task stencils within a graphical user interface for selection by a user to design a workflow (para [0005], [0011]-[0012], [0046], [0048]), each task stencil from the plurality of task stencils representing at least one task (para [0008], [0011]-[0012], [0041], Fig. 3), receiving from a user a selection of a task stencil from the plurality of task stencils (para [0011]-[0012], [0042]), receiving from a user a selection of a plug-in module from a plurality of plug-in modules (para [0010], [0012]-[0014], [0046], [0049]), each plug-in module representing a corresponding network device on a computing network (para [0014], [0058] - "a system work item can be configured to perform an action that may be implemented on a source network device (or downstream device) that caused an incident"; e.g. work items may correspond with, or are associated with, network devices), one or more activation commands for activating (para [0050]) a corresponding network device, of a plurality of network devices connected to a computer network (para [0027], [0058]), associating an activation command of a plug-in module with a task (para [0014], [0052]), and a task/action to be performed/executed in response to a security event (para [0009], [0028], [0052]).

Thus, the inventions listed as Groups I-III lack unity of invention because they do not share a same or corresponding special technical feature providing a contribution over the prior art.