

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 August 2007 (02.08.2007)

PCT

(10) International Publication Number
WO 2007/087074 A2

(51) International Patent Classification: **Not classified**

(21) International Application Number:
PCT/US2006/049552

(22) International Filing Date:
28 December 2006 (28.12.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/339,127 25 January 2006 (25.01.2006) US

(71) Applicant (for all designated States except US): **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **SNOVER, Jeffrey P.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **PAYETTE, Bruce, G.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **HUANG, Dana, Jin**; One Microsoft Way, Redmond, WA 98052-6399 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI,

GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

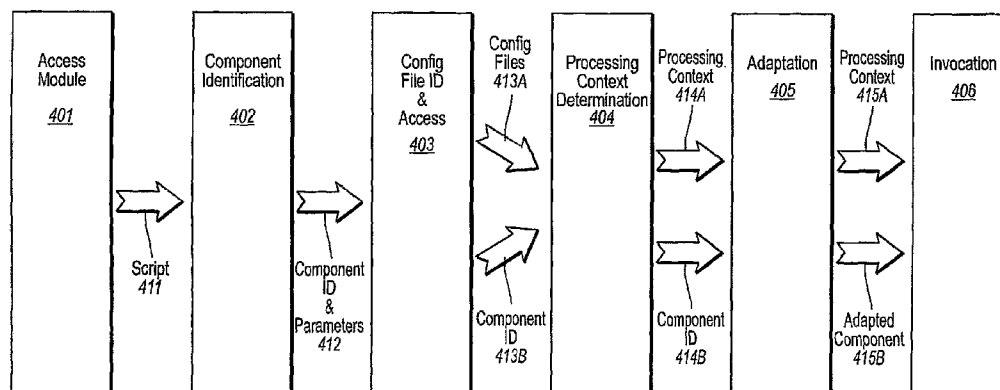
Published:

- without international search report and to be republished upon receipt of that report

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

(54) Title: EXTERNAL CONFIGURATION OF PROCESSING CONTENT FOR SCRIPT

400



(57) Abstract: Script is accessed and interpreted to identify an executable component. Processing context configuration files may then be used to identify an appropriate processing context for the identified executable component. Examples of processing context include, but are not limited to, a process in which the identified executable component is to run, one or more adaptations to perform on the component prior to running, and/or a security context in which to run the component. Processing context thus need not be specified in the actual script itself. The identified components may then be executed in the identified processing context.

EXTERNAL CONFIGURATION OF PROCESSING CONTENT FOR SCRIPT

BACKGROUND

Background and Relevant Art

5 [0001] Computing systems have revolutionized the way we work and play. Computing systems come in a wide variety of forms including laptop computers, desktop computers, personal digital assistants, telephones, and even devices that have not been conventionally associated with computing systems such as, for example, refrigerators and automobiles. Computing systems may even comprise a number of
10 constituent computing systems interconnected via a network. Thus, some computing systems may be small enough to fit in the palm of the hand, while others are spread over much of the globe.

[0002] Regardless of their physical form, computing systems are composed of hardware and software. The hardware includes most fundamentally at least one
15 processor and memory. The software includes instructions that may be embodied in the memory or in storage, and that can be accessed and executed by the processor(s) to direct the overall functionality of the computing system. Thus, software is critical in enabling and directing the functionality of the computing system.

[0003] The software in a typical computing system will typically include an
20 operating system and application programs. The operating system typically provides the core functionality common across multiple application programs. For instance, the operating system provides Application Program Interfaces (often termed "APIs") that provide underlying file systems, memory management, security, networking, user interfacing, and other core functionality to application programs. The operating
25 system also initiates, manages, and terminates multiple processes on a single computing system.

[0004] A "process" is a term of art that is used to describe a virtual address space that includes a collection of resources that may be shared by one or more running executable components that are included in that process. The resources may include a process identifier, one or more execution threads, file handles, shared memory, and
5 shared processor time. A process may also impose constraints on the executable component(s) that are run in that process so that order may be properly maintained. For instance, a process may expect a data structure or object of a particular type to have a specific structure, and may require that each component executing in the process use memory in a consistent manner.

10 [0005] "Script" is a term used to describe a sequence of commands that may be interpreted to form computer-executable instructions during run-time immediately before the computer-executable instructions are actually executed by the processor(s). Often, the commands will be used to execute specific components. The components invoked by the script are run in a certain processing context that is implied or
15 expressed in the script that invokes the component. The processing context may include the process that the component runs in and the security context in which the component is run. For instance, the security context may specify that the security mechanism for the component is to treat the user as a particular entity, and/or to run the component on a particular machine. It may be needful or advantageous for
20 components to be run in a particular processing context.

[0006] As a specific example, by default, the script may invoke a component that is run within a particular process. However, the component may not be compatible with the current process. For instance, the component may rely on a functions library that is not available to the process, or perhaps the component may not function as
25 intended within the context of that process. Alternatively or in addition, the script

may have been drafted by an author that is not trusted within the context of that process. In any of these cases, the script language may be altered to specify that the component is to be run in a different process. Furthermore, if the component is to be run outside of a default security context, that security context would be identified as well in the script language.

[0007] Thus, whenever a processing context of a script component is outside of the default processing context, the script is changed as well to reflect the new processing context. Altering the script in this manner can be a cumbersome process.

BRIEF SUMMARY

[0008] Script is accessed and interpreted to identify an executable component. Processing context configuration files may then be used to identify an appropriate processing context for the identified executable component. Examples of processing context include, but are not limited to, a process in which the identified executable component is to run, one or more adaptations to perform on the component prior to running, and/or a security context in which to run the component. Processing context thus need not be specified in the actual script itself.

[0009] 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 as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

- [0010] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:
- 5 [0011] Figure 1 illustrates a suitable computing environment in which the principles of the present invention may be employed;
- [0012] Figure 2 illustrates a flowchart of a method for identifying a particular processing context for running script in a computing system;
- [0013] Figure 3 illustrates a flowchart of a method for identifying one or more processing context configuration files associated with the identified executable component;
- 15 [0014] Figure 4 illustrates a processing flow associated with the method for identifying a particular processing context of Figure 2;
- [0015] Figure 5A schematically illustrates processing in which multiple components of a script are run within a single process, but in which the processing context of the second components is changed to reflect that some adaptation of the second component is to occur prior to execution;
- 20 [0016] Figure 5B schematically illustrates processing in which multiple components of a script are run in separate processes, with potentially some pre-processing adaptation or other processing context to be enforced on one of the components prior to execution; and
- [0017] Figure 5C schematically illustrates processing in which multiple components are run in separate process, and in which processing control may pass between subsequent components in a child process without passing control back to a parent process.
- 25

DETAILED DESCRIPTION

[0018] The present invention extends to the identifying of a processing context associated with one or more components executed by a script, without having to refer to processing context identification within the script itself. Thus, if the processing context changes, the script itself need not change. Instead, one or more processing context configuration files associated with each component are referred to in identifying the processing context. Should the processing context for a script component change, the configuration files are simply altered, rather than changing any script that invokes that component.

10 [0019] First, an example computing system in which the principles of the present invention may operate will be described with respect to Figure 1. Then, the principles of the present invention will be described in further detail with respect to the subsequent Figures 2, 3, 4, 5A, 5B and 5C. The embodiments of the present invention may comprise a special purpose or general-purpose computer including various computer hardware, as discussed in greater detail below.

[0020] Figure 1 shows a schematic diagram of an example computing system 100 that may be used to implement features of the present invention. The described computing system is only one example of such a suitable computing system and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the invention be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in Figure 1.

[0021] Computing systems are now increasingly taking a wide variety of forms. Computing systems may, for example, be handheld devices, appliances, laptop computers, desktop computers, mainframes, or distributed computing systems. In this description and in the claims, the term "computing system" is defined broadly as

including any device or system (or combination thereof) that includes at least one processor, and a memory capable of having thereon computer-executable instructions that may be executed by the processor. The memory may take any form and may depend on the nature and form of the computing system. A computing system may be distributed over a network environment and may include multiple constituent computing systems.

[0022] Referring to Figure 1, in its most basic configuration, a computing system typically includes at least one processing unit 102 and memory 104. The memory 104 may be volatile, non-volatile, or some combination of the two. An example of volatile memory includes Random Access Memory (RAM). Examples of non-volatile memory include Read Only Memory (ROM), flash memory, or the like. The term "memory" may also be used herein to refer to non-volatile mass storage. Such storage may be removable or non-removable, and may include (but is not limited to) PCMCIA cards, magnetic and optical disks, magnetic tape, and the like.

[0023] As used herein, the term "module" or "component" can refer to software objects or routines that execute on the computing system. The different components, modules, engines, and services described herein may be implemented as objects or processes that execute on the computing system (e.g., as separate threads) as part of a protocol. While the system and methods described herein may be implemented in software, implementations in hardware, and in combinations of software and hardware are also possible and contemplated.

[0024] In the description that follows, embodiments of the invention are described with reference to acts that are performed by one or more computing systems. If such acts are implemented in software, one or more processors of the associated computing system that performs the act direct the operation of the computing system in response

to having executed computer-executable instructions. An example of such an operation involves the manipulation of data. The computer-executable instructions (and the manipulated data) may be stored in the memory 104 of the computing system 100.

5 [0025] Computing system 100 may also contain communication channels 108 that allow the computing system 100 to communicate with other computing systems over, for example, network 110. Communication channels 108 are examples of communications media. Communications media typically embody computer-readable instructions, data structures, program modules, or other data in a modulated data
10 signal such as a carrier wave or other transport mechanism and include any information-delivery media. By way of example, and not limitation, communications media include wired media, such as wired networks and direct-wired connections, and wireless media such as acoustic, radio, infrared, and other wireless media. The term computer-readable media as used herein includes both storage media and
15 communications media.

[0026] Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer. By way of
20 example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose
25 computer. When information is transferred or provided over a network or another

communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included
5 within the scope of computer-readable media.

[0027] Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Although the subject matter has been described in language specific to structural
10 features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

[0028] Figure 2 illustrates a flowchart of a method 200 for identifying a particular
15 processing context for running script in a computing system. The computing system may be structured as described above for the computing system 100 of Figure 1, although any computing system that is capable of executing script and otherwise being adaptable to perform the principles of the present invention will suffice. Figure 4 illustrates a processing flow 400 including various components and data that may be
20 used to implement the method 200 of Figure 2. Accordingly, the method 200 of Figure 2 will be described with frequent reference to the processing flow 400 of Figure 4.

[0029] First, the script to be executed is accessed (act 201). The script may be accessed by receiving the script from another computing system, or by accessing the
25 script from a source internal to the computing system. For instance, if the method 200

is performed in the context of the computing system 100 of Figure 1, the script may be accessed from the memory 104. Referring to Figure 4, an access module 401 accesses script 411 and provides the script 411 to components further along the processing flow 400. Mechanisms for accessing script are known in the art and thus will not be described in detail here. The script 411 may include a script command line, multiple script command lines and/or perhaps even just a portion of one or more command lines. For instance, the script 411 may be just a portion of a script pipeline command.

[0030] Next, the script is interpreted (act 202) to identify one or more executable components to invoke (act 203). There are a number of conventional ways to interpret script to thereby identify one or more components to invoke. The principles of the present invention are consistent with any of those conventional methods and are also likely compatible with script interpretation technology that is yet to be developed so long as that script interpretation allows for the identification of components being invoked by that script. Referring to Figure 4, the component identification module 402 receives control of the script 411 from the script access module 401 to permit the component identifiers and associated parameters (collectively referred to as element 412) to be provided further along the processing flow 400.

[0031] Referring back to Figure 2, for each identified executable component, a functional, result-oriented step for discovering a processing context in which the script should be executed at least based on one or more processing context configuration files (step 210) is performed. While this step may be performed by using any combination of corresponding acts that accomplished this purpose, the step 210 is illustrated as being accomplished using constituent acts 211 and 212 in Figure

[0032] Specifically, one or more processing context configuration files associated with the identified executable component are identified (act 211). Figure 3 illustrates one contemplated method 300 for identifying one or more processing context configuration files associated with an identified executable component. A correlation is identified between one or more properties of the identified executable component and one or more configuration files (act 301). Then, the correlated one or more configuration files are identified as being the one or more processing context configuration files (act 302).

[0033] For instance, perhaps the name of the executable components alone is sufficient to identify one or more associated processing context configuration files. Component dependencies may also be useful in identifying associated processing context configuration files. For instance, a component may depend on a particular functions library for proper execution. The author, creation date, version number, and the like may also be relevant properties used to identify one or more associated configuration files. Referring to Figure 4, the processing context identifying and access module 403 uses the component identifier and parameter(s) provided by the component identification module 402 to provide the configuration files 413A and associated component identifier 413B (which in some embodiments may be included within the configuration file(s)) further down the processing flow 400.

[0034] The processing context configuration files are then used to identify the processing context of the associated component. The processing context of the component may include, for example, any one or more of the following: an identification of a process in which to run the identified executable component, one or more adaptations to perform on the identified executable component prior to being run (e.g., change the name or type of a field, or perform some calculation), a security

context in which to run the identified executable component (e.g., an identification of a user security context, or an identification of a machine on which to run the component). Referring to Figure 4, the processing context determination module 404 uses the configuration files 413A and component identifiers 413B to identify the processing context 414A for each component.

[0035] Several concrete examples will now be provided to clarify the principles more generally described above. In a first example, consider a script that includes the sequential execution of three components, C1 followed by C2 followed by C3. This may be represented by the following sequence: C1|C2|C3. Now suppose that there has been some change to component C2 which no longer makes it advisable or possible to have the component run in the same process as components C1 and C3. In that case, the processing context configuration file might read as follows for component C2 if represented in one eXtensible Markup Language (XML) format.

```
<CMD Name = "C2">  
    <OutOfProcess>True</OutOfProcess>  
</CMD>
```

[0036] The use of such a configuration file makes it much easier to change the processing context of the configuration file. For instance, if the component C2 were changed such that it is once again advantageous to run the component C2 in the same process with components C1 and C3, the processing context configuration file may once again be changed to reflect this change as opposed to changing all script that references component C2. For instance, the configuration file may be changed to read as follows:

```
<CMD Name = "C2">  
    <OutOfProcess>False</OutOfProcess>  
</CMD>
```

5

[0037] In this example, the configuration file is identifiable by the name of the component. However, as previously mentioned, the configuration file(s) may be identified by other properties of the component such as author, creation date, or others.

10 [0038] As mentioned above, the processing context may involve much more than whether or not the component is run in-process or out-of-process. Take, for example, the following configuration file specifying a far more complex processing context for component C2:

```
15 <CMD Name = "C2">  
    <Version>V2</Version>  
    <Adaptor>LMD=>MD</Adaptor>  
    <OutOfProcess>True</OutOfProcess>  
    <RunAsUser>Bob</RunAsUser>  
20 <HostComputer>Bobs</HostComputer>  
    </CMD>
```

[0039] Here, the processing context configuration file for component C2 specifies what version of interpreter is required to run in the process. Thus, if the component
25 C2 requires a different version of the interpreter than components C1 and C3, the

component C2 would be run out-of-process unless different versions of the interpreter were possible in the same process.

[0040] The "Adaptor" element specifies an adaptation that is to be performed prior to execution of the component. Here, the LMD (Last Modified Date) Field is changed to the MD (Modified Date) field. This allows the component C2 to be modified in a manner that allows the component to be executed within the component execution sequence. Referring to Figure 4, the adaptation component 405 may modify the component identified by the component identifier 414B according to any adaptations specified in the processing context 414A. The adaptation module 405 then provides the processing context 415A and the adapted component 415B to the invocation component 406.

[0041] The "OutOfProcess" field specifies that the component C2 is to be run out-of-process. The "RunAsUser" field specifies that the component C2 is to have the same security context as is permissible should the identified user be making the same requests. In the example case, the system will permit all operations being performed by component C2 so long as the system would permit that operation if requested by Bob.

[0042] The "HostComputer" field specifies the host computer on which the component C2 is to be executed. In this case, the component C2 is to run on the host computer identified as "Bobs". As previously mentioned, the execution is performed using the security context for Bob on that host computer since the "RunAsUser" field specifies "Bob".

[0043] Referring back to Figure 2, the invocation component then invokes the identified executable component in the identified processing context (act 213). This invocation may be performed on the same computing system as or on a different

computing system than the computing system that interpreted the script. For instance, in the above example, if the host computer "Bobs" is not the same computer that runs the invocation module 406, then the invocation module 406 may perform any actions helpful to remote the component C2 on the proper host computer identified as "Bobs".

5 Whether the component is remotely invoked or not, the invocation module 406 may return the results of the component C2 execution.

[0044] Figure 5A illustrates an example in which all three components C1, C2 and C3 are run in the same process, but with component C2 being adapted to be compatible with running in the same process as components C1 and C3. Each process instantiates its own interpreter for interpreting the script.

10 Figure 5B illustrates an example in which component C2 is run out-of-process. In this case, the component C2 may also be adapted as specified in the configuration file.

[0045] Figure 5C illustrates another example in which four components are executed in sequence as represented by the sequence C1|C2|C3|C4. In this case, components C1 and C4 are run in Process A, whereas components C2 and C3 are run in Process B. In this case, the processing context determination module 404 may use the processing context configuration files for components C2 and C3 to identify that the components should not be run in-process within Process A. The processing context determination module 404 then makes another determination as to whether or not the components C2 and C3 are compatible such that they may run in the same process outside of Process A. For instance, if components C2 and C3 depend on different functions libraries that cannot be run in the same process, components C2 and C3 would have to run in different processes outside of Process A.

[0046] In this case, however, the processing context determination module 404 has decided that components C2 and C3 can run in the same process within Process B. In such a case, there are processing efficiencies since processing control does not need to be returned from component C2 back to Process A, before passing processing control back to Process B for execution of component C3. Instead, execution of component C2 may smoothly transition to execution of component C3 within the same Process B.

[0047] Accordingly the principles of the present invention provide a flexible mechanism for specifying and altering the processing context in which a script component is to be executed, without requiring that the script itself be altered to specify the processing context.

[0048] The present invention may be embodied in other specific forms without
5 departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

What is claimed is:

1. A method for identifying a particular processing context for running script in a
5 computing system, the method comprising:
an act of interpreting script to identify an executable component to invoke;
an act of identifying one or more processing context configuration files associated with the
identified executable component; and
an act of using the one or more processing context configuration files to identify a processing
10 context for the identified executable component.
2. A method in accordance with Claim 1, wherein the processing context includes an
identification of a process in which to run the identified executable component.
- 15 3. A method in accordance with Claim 1, wherein the processing context includes one
or more adaptations to perform on the identified executable component prior to being run.
4. A method in accordance with Claim 1, wherein the processing context includes a
security context in which to run the identified executable component.
20
5. A method in accordance with Claim 4, wherein the security context includes an
identification of a user security context.
6. A method in accordance with Claim 1, wherein the act of interpreting is performed
25 by an executable interpretation component.
7. A method in accordance with Claim 1, wherein the act of identifying one or more
processing context configuration files associated with the identified executable component comprises:

an act of identifying a correlation between one or more properties of the identified executable component and one or more configuration files; and

an act of identifying the correlated one or more configuration files as being the one or more processing context configuration files.

5

8. A method in accordance with Claim 7, wherein the one or more properties includes at least a name of the identified executable component.

9. A method in accordance with Claim 7, wherein the one or more properties includes at
10 least a dependency of the identified executable component.

10. A method in accordance with Claim 1, further comprising:
an act of invoking the identified executable component in the identified processing context.

11. A method in accordance with Claim 10, wherein the act of invoking the identified
15 executable component in the identified processing context is performed on a computing system that is different than the computing system that performed the act of interpreting.

12. A method in accordance with Claim 10, wherein the act of invoking the identified
20 executable component in the identified processing context is performed on a computing system that is the same as the computing system that performed the act of interpreting.

13. A computer program product comprising one or more computer-readable media having thereon computer-executable instructions for perform a method for identifying a particular processing context for running script in a computing system, the computer-executable instructions comprising:

- 5 at least one computer-executable instruction for accessing script;
- at least one computer-executable instruction for interpreting the script to identify an executable component; and
- at least one computer-executable instruction for discovering a processing context in which the script should be executed at least based on one or more processing context configuration files.

10

14. A computer program product in accordance with Claim 13, wherein the processing context includes at least one of the following:

- an identification of a process in which to run the identified executable component;
- one or more adaptations to perform on the identified executable component prior to being run;
- 15 or
- a security context in which to run the identified executable component.

15. A computer program product in accordance with Claim 13, wherein the processing context includes at least two of the following:

- 20 an identification of a process in which to run the identified executable component;
- one or more adaptations to perform on the identified executable component prior to being run;
- or
- a security context in which to run the identified executable component.

25 16. A computer program product in accordance with Claim 13, wherein the computer-executable instructions further comprise:

- at least one computer-executable instructions for invoking the identified executable component in the discovered processing context.

17. A computer program product in accordance with Claim 13, wherein the one or more computer-readable media are physical memory and/or storage media.

18. One or more computer readable media having thereon computer-executable instructions that, when executed by a processor of a computing system, cause the computing system to instantiate the following in system memory of the computing system:

an interpretation component configured to access and interpret script to identify an executable component; and

a processing context determination module configured to using one or more processing context configuration files correlated to the identified executable component to thereby identify a processing context for the identified executable component.

19. One or more computer-readable media in accordance with Claim 18, wherein the processing context includes at least one of the following:

an identification of a process in which to run the identified executable component;
one or more adaptations to perform on the identified executable component prior to being run;

or

a security context in which to run the identified executable component.

20. One or more computer-readable media in accordance with Claim 18, wherein the one or more computer-readable media are physical memory and/or storage media.

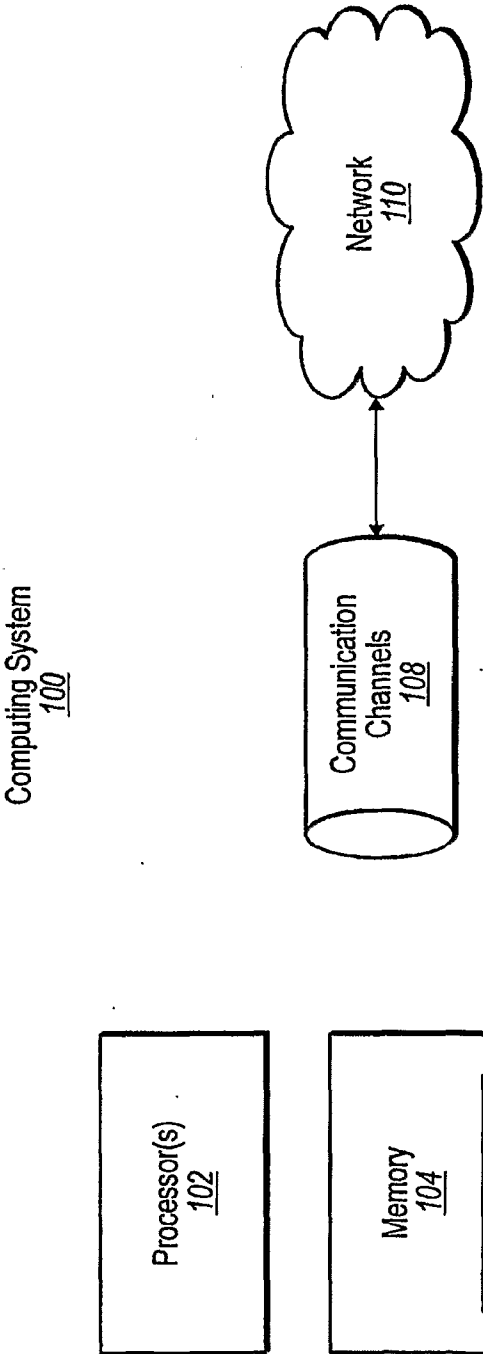
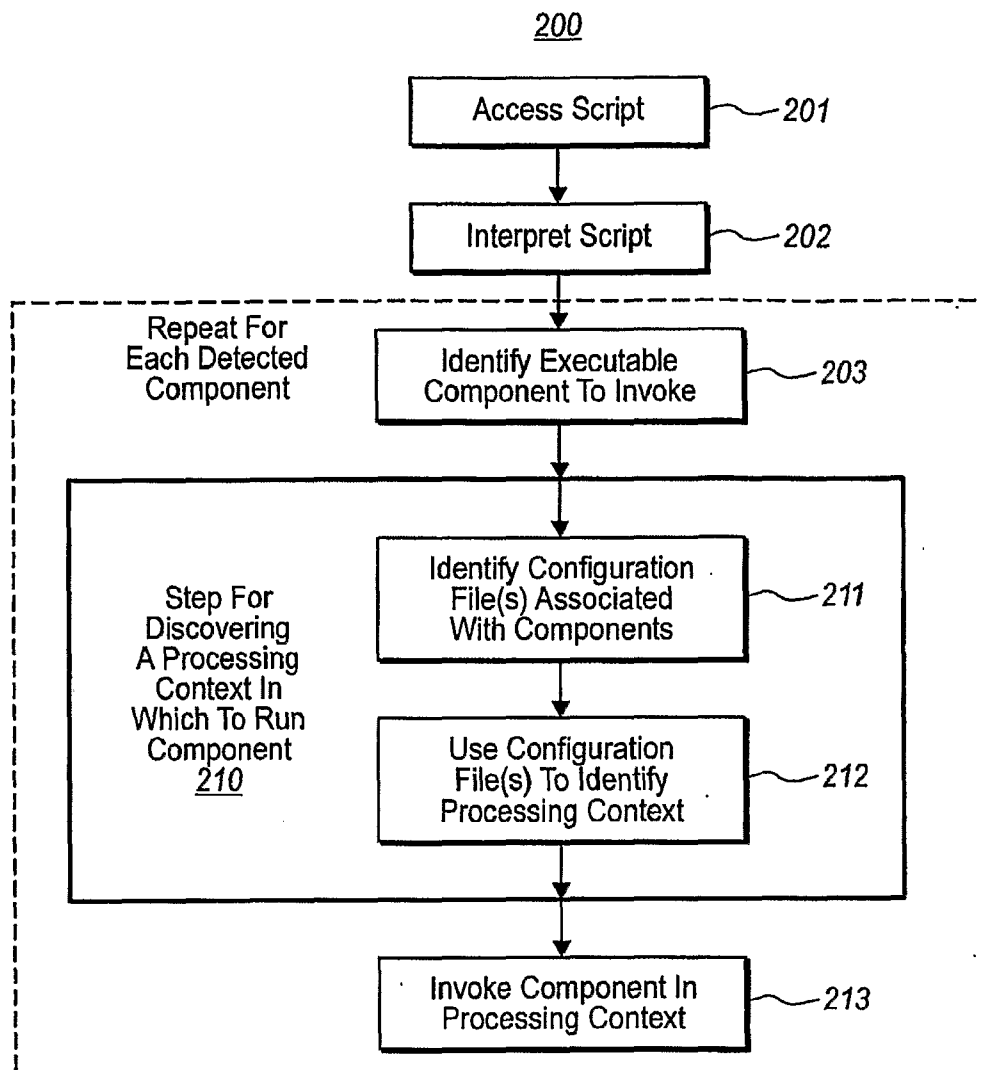
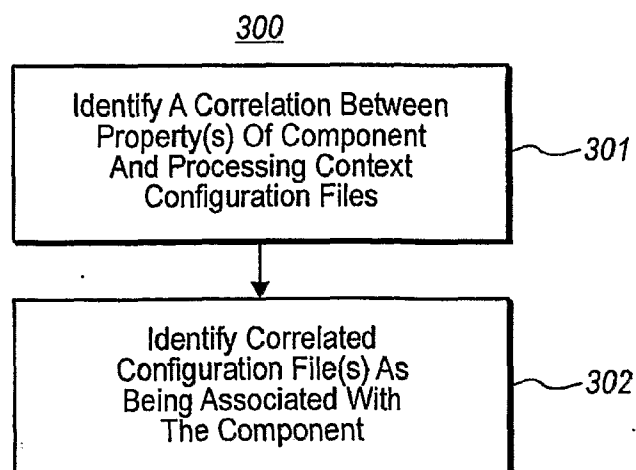


FIG. 1

2/4

**FIG. 2****FIG. 3**

400

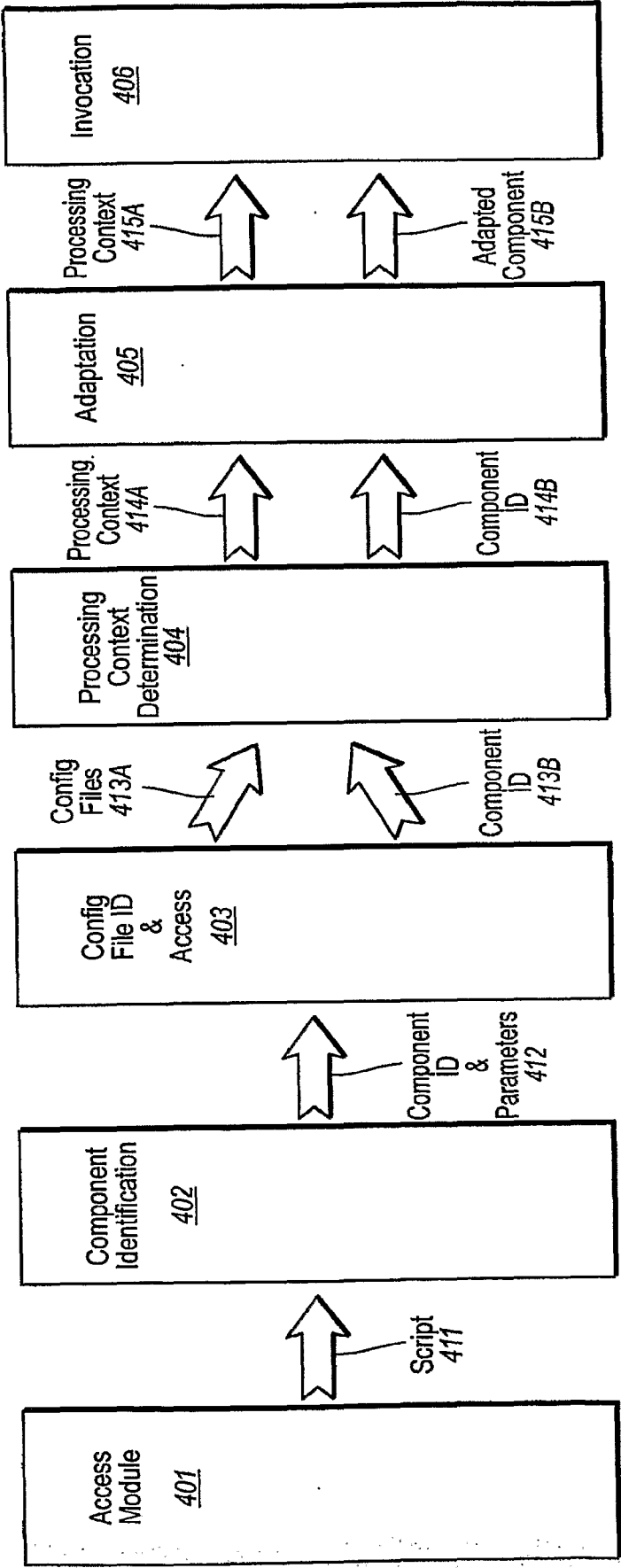


FIG. 4

4/4

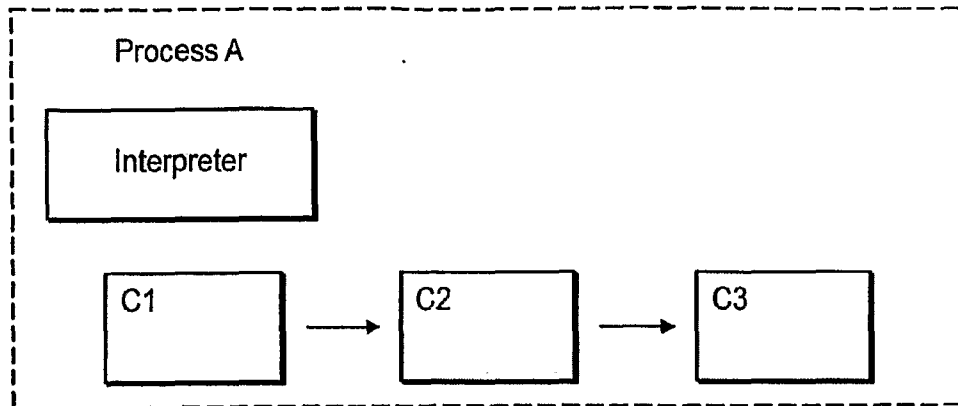


FIG. 5A

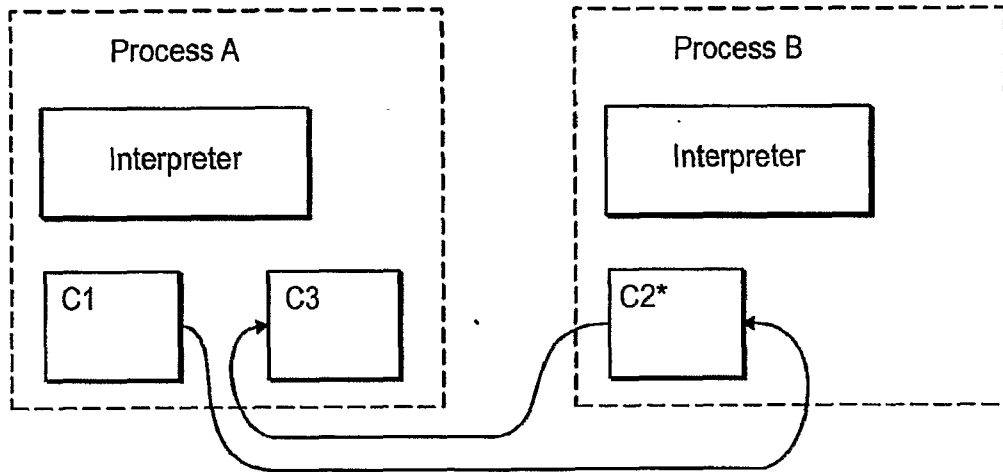


FIG. 5B

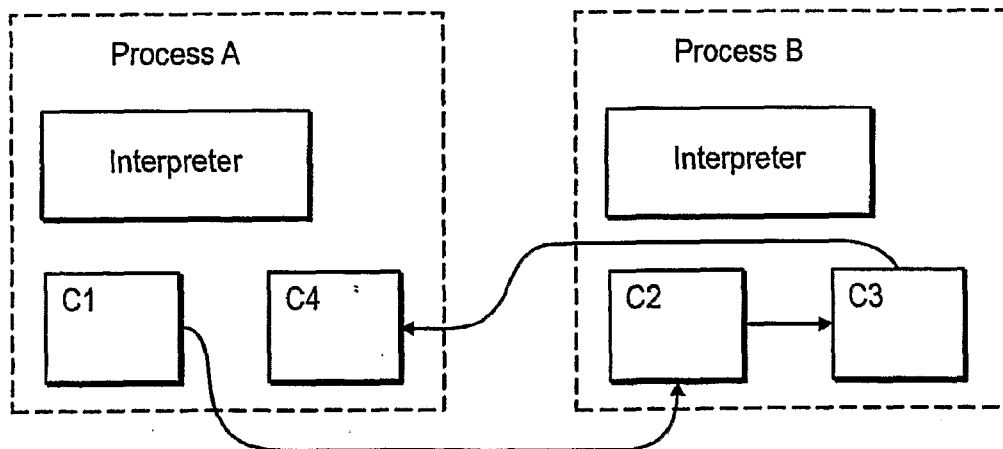


FIG. 5C