

**422.** If the system setting operation will have an impact on the system setting, the operating system performs a mitigated system setting operation at **420** according to a mitigation strategy such as illustrated in **FIG. 3**. The change to the system setting, if any, is recorded in a log at **415**.

[**0059**] If the operation represents a request to load an extension to the operating system at **424**, the operating system determines if the application program (e.g., xxxx.exe) desires protection (e.g., to enable an “undo”) at **426**. For example, the application program may explicitly inform the operating system of a desire for protection. If the application program does not want protection, the operating system allows the extension to load at **428**. If the application program indicates that protection is desired, the operating system determines if the extension is a foreign extension (e.g., supplied by a third party) at **430**. If the extension is not foreign, the operating system allows the extension to load at **428**. If the extension is foreign, the operating system performs a mitigated extension load at **432** according to a mitigation strategy such as illustrated in **FIG. 3**. The extension load may be recorded in a log. For example, the recording may be configurable by a user of the computing system executing the operating system.

[**0060**] With virtualization, an application creates and modifies objects in their own local namespace, while the operating system creates and modifies objects in the global namespace. There is one global namespace, and potentially multiple local namespaces. For create operations, the application creates the object in its local namespace. When an application attempts to modify an object, the operating system checks if the object resides in the local namespace for the application. If the local object exists, the application opens the object in its local namespace. If the application attempts to modify an object in the global namespace, the operating system copies the object into the application’s local namespace and allows the operation to occur on that local object. If the resource does not exist in the local or global namespace, the open operation fails.

[**0061**] Referring next to **FIG. 5**, an exemplary flow chart illustrates operation of a method of providing access control for system settings. Even though **FIG. 5** illustrates an example related to system settings, the virtualization aspect of the invention may be utilized for other objects (e.g., named objects) and namespaces. In **FIG. 5**, an embodiment of the invention such as an operating system analyzes an operation on a system setting requested by, for example, an application program. In particular, the operating system determines if the requested operation will write or delete a system setting at **502**. If the requested operation will not write or delete a system setting (e.g., read-only access is requested), the operating system determines if a virtual copy of the system setting currently exists at **504**. If a virtual copy exists, the operating system identifies the virtual copy at **506** and performs the requested operation on the virtual copy of the system setting at **508**. If a virtual copy does not exist, the operating system performs the requested operation on the system setting at **508**.

[**0062**] If the requested operation will write or delete a system setting, the operating system determines if the requesting application program is associated with a read-only key (e.g., the requesting application program is not a trusted installer) at **510**. If the requesting application pro-

gram is associated with read-only access (e.g., via an access control list maintained by the operating system), the operating system will fail or deny the requested operation at **512**. If the requesting application program is not associated with a read-only access, the operating system determines if the requested operation will write or delete a system-restricted setting at **514**. If the requested operation will write or delete a system restricted setting, the operating system determines if the requesting application program is approved to perform the operation at **516**. For example, the operating system may determine if the requesting application program has administrator privileges on the computing system. If the requesting application program is approved to perform the operation, the operating system will perform the requested operation at **508**. If the requesting application program is not approved to perform the operation, the operating system will fail or deny the requested operation at **512**.

[**0063**] If the requested operation will not write or delete a system restricted setting, the operating system determines if the requested operation is for a protected setting (e.g., a copy of a system setting associated with the requesting application program) at **518**. If the operating system determines that the requested operation is for a protected setting, the operating system virtualizes the protected setting by the application identifier of the requesting application program at **520**. That is, the operating system identifies the virtual copy of the system setting and performs the requested operation on the identified, virtual copy of the system setting at **508**. If the operating system determines that the requested operation is not for a protected setting, the operating system determines if the requested operation is for a private setting (e.g., a system setting associated with the requesting application program) at **522**. If the operating system determines that the requested operation is for a private setting, the operating system performs the requested operation on the private system setting at **508**. If the operating system determines that the requested operation is not for a private setting, the operating system ends processing and fails the request silently or explicitly.

[**0064**] When the application attempts to delete an object from the local namespace and a global object with the same name exists, the system marks the local object as deleted but leaves that object in the namespace. Thus, the system is able to detect that the application’s queries for that object should not see that object’s name. When the application attempts to delete an object that exists in the local namespace but not the global namespace, the system deletes the local object. Depending on the operating system configuration, deleting a global object may result in deleting all the corresponding local objects. The system may allow the application to designate whether their corresponding objects should get deleted in this manner, and the resource provider stores that designation on the local object. Also, adding a global object may result in deleting all the corresponding objects marked as deleted from all local namespaces.

[**0065**] With this design, the application thinks that it is working in the global namespace, but in reality, it works in its own namespace. The system handles full path queries, enumerations, and other operations to make the application think that it is working in the global namespace. For example, namespace enumeration includes listing all files under a particular directory. The system queries all the objects in the specified namespace (e.g., starting first with