

structure of data objects, such as data objects **204**, **206**, **208** and **210**. Typically, the XML Store **202** also provides the overall structure in which objects are named, stored and organized. Additionally, the store provides the protocols for accessing any object within the store **202**. Although shown in relation to an XML store, other data object configurations or collections may incorporate the aspects of the present invention. Data objects **204**, **206**, **208** and **210** are XML data objects that represent actual file-type data. The objects **204**, **206**, **208** and **210** may be accessed and/or modified by a user or another program module. Of course, the XML Store **202** may comprise many other objects as indicated by ellipses **223** and **225**.

[0034] Typically, each data object **204**, **206**, **208** and **210** has some form of meta information object (not shown) that is associated with each object, the meta information comprises information such as the author of the object, the time the object was last accessed, among others. This meta information may be stored as part of the data object or as part of another object having a pointer or some other identifying element that associates the meta information object with its particular data object.

[0035] In addition to the meta information objects, a data object may also be associated with a version-specific property object, such as objects **220** and **222**. Version specific properties **220** and **222** are associated with data objects **208** and **210**, respectively. The version-specific properties **220** and **222** comprise version-specific information and may be invalidated by other events occurring with respect to the objects **208** and **210** respectively.

[0036] The software environment **200** shown in FIG. 2 also illustrates the interaction of the XML Store **202** and application programs, such as applications **224** and **226**. In one embodiment, application program **226** is a client application program that operates on a client system apart from a server system, which is the location of the XML Store **202**. In other embodiments, the application program, i.e., program **224** may actually be part of the server system. Applications **224** and **226** interact with the XML store **202** through application program interfaces **228** and **230**, respectively. Additionally, the application interfaces **230** may actually communicate with the XML store **202** through a complex network, such as the Internet **232**, according to predefined protocols. Importantly, in the present invention an application or program module, such as applications **224** and **226**, communicates with the XML store **202** in one way or another, to access data objects, such as objects **204**, **206**, **208** and **210** wherein the access may involve moving, copying, deleting, reading, executing or updating the object, among others.

[0037] Application programs **224** and **226** may access the objects **204**, **206**, **208** and **210** through a layer of application modules, i.e., the services layer **236**. The services layer **236** may provide various functions, such as ensuring that an object access request relates to an existing object, whether the module making the request has permission to make and perform the request, among others. This layer of interaction **236** is in addition to possible application program interfaces **228**, **230** and possible operating system interfaces (not shown) that may be part of the client or server computer systems.

[0038] With respect to the version-specific properties **220** and **222**, in an embodiment of the invention, application

programs **224** and **226** may create and use the version-specific properties **220** and **222** related to objects **208** and **210** respectively. Alternatively the services layer **236** may create and use the version-specific properties. Once a version-specific property has been created, another application may access the property and decide to perform an operation on the object based on the evaluation of the version-specific property **220** or **222**. Additionally, the other applications may perform actions on a data object that may invalidate the version-specific property thereby causing a different result once an application tests for the existence of a valid version-specific property.

[0039] In one particular example, the services layer **236** provides a virus-scan function that performs virus scanning and cleaning functions each time an object, e.g., objects **204**, **206**, **208** or **210**, is accessed by any other application or module. To further expand the example, the application program **224** may be a word processing application and the objects **204**, **206**, **208** and **210** are word processing type objects such as XML objects having specific text components. In such a case, the virus scan program module, as part of services layer **236**, may actually be utilized to scan objects that the word processing application **224** requests to access. In this example, the virus scanner may create version-specific properties, such as properties **220** and **222** for objects that have been scanned for viruses. Therefore, the next time a request is made by application **224** for one of the objects **208** and **210**, the virus scan application merely identifies the existence of a valid version-specific property, such as property **220** and **222**, to determine whether another scan operation is necessary. If a valid version-specific property is identified, then no scan is necessary in this particular example. Contrarily, if no version-specific property is identified, such as when object **204** or **206** are accessed, then the virus scan application recognizes that these objects have either not been scanned or have been modified since the time when they were scanned.

[0040] Continuing with the virus-scan example, assuming virus scan application, as part of services layer **236**, scans one of the objects **204** or **206**, a new version-specific property (not shown) may then be created and associated with the scanned object. The newly created property (not shown) is then stored along with the object so that the property is available for future access requests.

[0041] In another embodiment, the version-specific property may be encoded with a digital signature that may be accessed and evaluated by other applications. This digital signature can then be tested to determine whether the file is a valid copy. In such a case, if the file has been tampered with by another application, i.e., corrupted, the digital signature will be invalidated. An invalidated digital signature may be treated as if the signature did not exist and, consequently, the data object may be treated as invalid.

[0042] The version-specific attribute may further comprise other security information to prevent unauthorized access to the attribute or the data object. This information is used by the services layer **236** as a means to lock the property from use by inappropriate applications, such as virus applications. The services layer may be configured to evaluate the version-specific property on each access for the purposes of ensuring that only valid applications access the property, or the data object itself.