

second alternative mode), an administrator configuration may be present that is capable of overriding the other configuration version binding information. Some or all of the various configuration data structures (e.g., the publisher configurations) themselves may be wrapped as assemblies, thereby benefiting from the characteristics of assemblies, including versioning of configurations, strong naming of configurations, and so on.

[0010] In this manner, the present invention enables applications to explicitly use different versions of assemblies from what the application as originally shipped had specified. This allows for exact management and control of assemblies during the lifecycle of the application. To determine the correct version, at runtime, in the first mode, the present invention first interprets the application manifest (e.g., released with the application), followed by a publisher configuration, if present, that may redirect (re-map) any assembly versions specified in the application manifest to other assembly versions. Then, if an application configuration is present, the application configuration is interpreted to redirect some or all of the current binding information, e.g., for the current assembly version to another assembly version, as specified therein.

[0011] In the second alternative mode, the present invention first interprets the application manifest (e.g., released with the application), followed by the application configuration, if present, that may redirect (re-map) any versions specified in the application manifest to other versions. Then, if a publisher configuration is present, and the application configuration does not bypass the publisher configuration via a special safe mode, the publisher configuration is interpreted to (possibly) redirect the current binding information, e.g., for that assembly version to another assembly version, as specified therein. Lastly, any administrator configuration is interpreted to (possibly again) change the bindings to assembly versions.

[0012] For efficiency, the present invention may build tables in an activation context in a pre-execution initialization phase to maintain the version mapping information, rather than interpreting the manifest and any configurations each time an assembly is needed, (i.e., per-request file mapping including adjusting for configurations is straightforward to implement, but less efficient). The activation context tables provide fast mapping from assembly names provided by the application, including version-independent names, to the correct versions as specified in the manifest (normally fully-named assemblies) and altered by any configurations. Once built, the tables may be cached, e.g., such as in the first alternative mode for the time that the application instance runs (lifetime of the process), whereby the information therein is available as needed. In an alternative mode such as the second alternative mode, the tables or other binding data may be dynamically recalculated.

[0013] To use the tables, in the pre-application execution phase when creating a new process, the operating system checks for an application manifest in same file system directory as the calling executable. In the first alternative mode, when an application manifest exists, the operating system checks for an activation context for the application that was built from the manifest and configurations. If the activation context does not exist (for example this is the first time application has been executed), or it exists but is not

coherent with current configuration, a new activation context is created via the application manifest and configurations.

[0014] At runtime, when a program requests creation of a global object, the operating system automatically consults the activation context built from the application and configurations to locate and load the appropriate assembly version. The operating system also maps any uses of this named object to the appropriate version to allow for multiple versions of the code module to run simultaneously without interfering with each other. By the activation context built from the application manifest and the configurations, an application may be efficiently bound to specific assembly versions and thereby be isolated from assembly version changes.

[0015] At the same time, changes to the bindings are enabled via the configurations.

[0016] Other objects and advantages will become apparent from the following detailed description when taken in conjunction with the drawings, in which:

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0017] FIG. 1 is a block diagram representing a computer system into which the present invention may be incorporated;

[0018] FIGS. 2A and 2B are block diagrams generally representing alternative modes, respectively, for binding applications to assemblies, including manifests and configurations that are used to specify assembly version bindings in accordance with aspects of the present invention;

[0019] FIGS. 3A and 3B are block diagrams generally representing manifests and configurations for binding an application to a specified assembly version in alternative modes, respectively, in accordance with aspects of the present invention;

[0020] FIG. 4 is an example of information maintained within an activation context in accordance with an aspect of the present invention;

[0021] FIG. 5 is a block diagram generally representing various assemblies for utilizing an activation context at runtime to locate and load a particular version of a requested assembly version in accordance with an aspect of the present invention;

[0022] FIGS. 6-7 comprise a flow diagram representing general steps taken to initialize an activation context in a first alternative mode based on manifests and configurations in accordance with an aspect of the present invention;

[0023] FIG. 8 is a representation of a dependency graph useful in constructing the activation context in accordance with an aspect of the present invention;

[0024] FIG. 9 is a flow diagram representing general steps taken to utilize an activation context during runtime in accordance with an aspect of the present invention; and

[0025] FIGS. 10-11 comprise a flow diagram representing general steps taken to determine assembly versions in a second alternative mode based on manifests and configurations in accordance with an aspect of the present invention;