(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101836186 B
(45)授权公告日 2015.02.25

(21)申请号 200880112352.9
(22)申请日 2008.10.17
(30)优先权数据
11/875881 2007.10.20 US
(85)PCT国际申请进入国家阶段日
2010.04.20
(86)PCT国际申请的申请数据
PCT/US2008/080260 2008.10.17
(87)PCT国际申请的公布数据
WO2009/052346 EN 2009.04.23
(73)专利权人 思杰系统有限公司
地址 美国佛罗里达
(72)发明人 J·诺德 M·钦塔 D·霍伊
(74)专利代理机构 北京泛华伟业知识产权代理有限公司 11280
代理人 王勇
(51)Int.Cl.
G06F 9/445(2006.01)
(56)对比文件
WO 0125894 A1,2001.04.12, 全文.
WO 9847074 A1,1998.10.22, 全文.
WO 2006039239 A1,2006.04.13, 全文.
审查员 曹姝妹

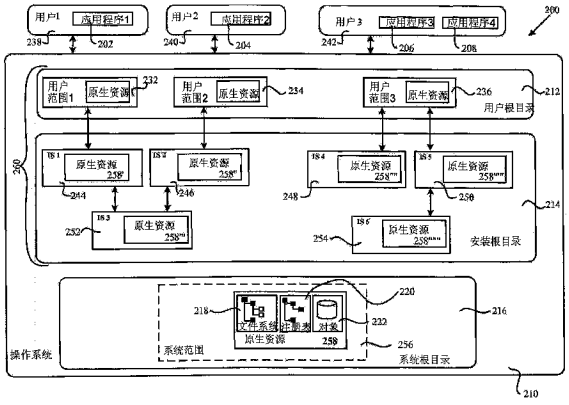
权利要求书2页 说明书69页 附图32页

(54)发明名称

用于在隔离环境之间通信的方法和系统

(57)摘要

用于在隔离环境中聚集各个安装范围的方法和系统，其中该方法包括首先限定用于包括各安装范围的聚集的隔离环境。在第一应用程序和第一安装范围之间建立第一关联。当第一应用程序为了正确执行而需要在该隔离环境中存在第二应用程序时，将所需的第二应用程序的映像装到第二安装范围上并且建立第二应用程序和第二安装范围之间的关联。在第一安装范围和第二安装范围之间建立关联，在第三安装范围中建立第三关联。第一关联、第二关联和第三关联的每一个被保存，并且在所限定的隔离环境中开始执行第一应用程序。



CN 101836186 B

1. 一种便于在安装范围之间传递资源信息的方法，该方法包括：

初始化包括第一用户根目录和第一安装根目录的第一隔离环境，所述第一用户根目录提供第一用户范围，所述第一安装根目录提供和第一应用相关联的第一安装范围；

创建虚拟范围，该虚拟范围代表所述第一安装范围与第二安装范围的聚集，所述第二安装范围和位于第二隔离环境中的第二应用相关联；

拦截来自第一应用的对原生资源的请求，其中，所述原生资源是以下中的至少一个：文件系统、注册表、系统对象、窗名称、用户专用的设置、用户专用的配置、当仅安装操作系统时呈现的资源；

在第一安装范围中执行对于与所请求的原生资源相对应的原生资源的实例的搜索；

当未能在第一安装范围中定位对应的原生资源时，则搜索第一安装范围和第二安装范围之间的关联，其中，通过所述虚拟范围来提供所述关联；

确定是否找到第一安装范围和第二安装范围之间的关联；

如果在第一安装范围和第二安装范围之间存在关联，则在和第二应用相关联的第二安装范围内搜索所请求的原生资源；

从所述第二安装范围获取与所请求的原生资源对应的所述原生资源的实例；以及

使用经由所述虚拟范围所获取的所述原生资源的实例来响应所拦截的请求。

2. 根据权利要求 1 所述的方法，其中，所述来自第一应用的请求是针对多个原生资源的请求。

3. 根据权利要求 1 所述的方法，还包括：由所述第一安装根目录提供第三安装范围。

4. 根据权利要求 3 所述的方法，还包括：所述第一安装范围具有比第三安装范围更高的优先级。

5. 根据权利要求 1 所述的方法，其中，所述第一应用在第一安装范围内执行，并且所述第二应用在第二安装范围内执行。

6. 根据权利要求 1 所述的方法，还包括：通过一种能拦截对原生资源的请求的过滤器驱动、驱动器或其他虚拟对象拦截来自所述第一应用的对原生资源的请求。

7. 一种便于在安装范围之间传递资源信息的系统，所述系统包括：

用于初始化包括第一用户根目录和第一安装根目录的第一隔离环境的模块，其中，所述第一用户根目录提供第一用户范围，所述第一安装根目录提供和第一应用相关联的第一安装范围；

用于创建代表所述第一安装范围与第二安装范围的聚集的虚拟范围的模块，其中，所述第二安装范围和位于第二隔离环境中的第二应用相关联；

用于拦截来自第一应用的对原生资源的请求的模块，其中，所述原生资源是以下中的至少一个：文件系统、注册表、系统对象、窗名称、用户专用的设置、用户专用的配置、当仅安装操作系统时呈现的资源；

用于在第一安装范围中执行对于与所请求的原生资源相对应的原生资源的实例的搜索的模块；

用于当未能在第一安装范围中定位对应的原生资源时，则搜索第一安装范围和第二安装范围之间的关联的模块，其中，通过所述虚拟范围来提供所述关联；

用于确定是否找到第一安装范围和第二安装范围之间的关联的模块；

用于如果在第一安装范围和第二安装范围之间存在关联，则在和第二应用相关联的第二安装范围内搜索所请求的原生资源的模块；以及

用于经由所述虚拟范围将所请求的原生资源从所述第二安装范围递送到第一应用的模块。

8. 根据权利要求 7 所述的系统，其中，所述来自第一应用的请求针对多个原生资源。

9. 根据权利要求 7 所述的系统，还包括：由所述第一安装根目录提供第三安装范围。

10. 根据权利要求 9 所述的系统，其中，所述第一安装范围具有比第三安装范围更高的优先级。

11. 根据权利要求 7 所述的系统，其中，所述第一应用在第一安装范围内执行并且所述第二应用在第二安装范围内执行。

12. 根据权利要求 7 所述的系统，还包括：用于通过一种能拦截对原生资源的请求的过滤器驱动、驱动器或其他虚拟对象拦截来自所述第一应用的对原生资源的请求的模块。

用于在隔离环境之间通信的方法和系统

技术领域

[0001] 本申请总的涉及应用程序之间的通信。更具体地，本申请涉及互相隔离并且位于一个隔离环境中的应用程序之间的通信。

背景技术

[0002] 用于减少关于应用程序的社交性 (sociability) 和兼容性问题的方案包括那些隔离应用程序执行的方案，这些方案使得应用程序能够访问到专用于执行该应用程序的用户和该应用程序这两者的唯一的一组原生资源。在一些方案中，在隔离期间要求另外的应用程序存在的应用程序被隔离在隔离环境中，该隔离环境包括用户执行的应用程序和当用户执行的应用程序运行时执行的另外的支持应用程序。每当主应用程序请求支持应用程序的帮助时，则创建一个包括主应用程序和支持应用程序的新的隔离环境。每当执行包括支持应用程序的实例的隔离环境时，重新产生支持应用程序所使用的资源。支持应用程序对资源所做的改变被限制到其中发生改变的隔离环境，并且当支持应用程序在不同的隔离环境中执行时，不保存所做的改变。支持应用程序对用户指定的且应用程序专用的资源做出的改变被保持在其中发生改变的隔离环境中，并且当支持应用程序在不同的隔离环境中执行时，不保存所做的改变。这些改变更适合专用于其中发生改变的隔离环境，并且应用程序设置也不在同一用户会话中执行的一个应用程序的多个实例间共享。

发明内容

[0003] 在一个方面中，示出并且描述了用于聚集隔离环境中各个安装范围的系统。该系统包括用于限定被用来包含各安装范围的聚集的隔离环境的装置。该系统还包括用于在第一应用和第一安装范围之间建立第一关联的装置。在建立第一关联之前，请求执行第一应用。该系统还包括用于确定和第一应用相关联的需求的装置，其中该需求标识出需要隔离环境中的第二应用存在。响应于该需求，所需的第二应用的映像被装到第二安装范围上以建立第二应用和第二安装范围之间的第二关联。该系统包括用于使第一安装范围和第二安装范围相关联的装置，并且建立在第一安装范围和第二安装范围的聚集和第三安装范围之间的第三关联。第一关联、第二关联和第三关联的每一个由系统保存，并且在隔离环境中由系统启动所请求的第一应用。

[0004] 在一个方面中，示出并且描述了用于从第一隔离环境中获取原生资源的系统。该系统包括用于拦截对原生资源的请求的装置。对原生资源的请求由和包括在第一隔离环境中的第一安装范围相关联的第一应用程序所产生。使用系统所提供的装置在第一安装范围中执行对于与所请求的原生资源相对应的原生资源的搜索。当未能在第一安装范围中定位对应的原生资源时，使用系统提供的装置做出关于在第一安装范围和第二安装范围之间是否存在关联的决定。第二安装范围可以代表第二应用的映像，其中，第二应用位于第二隔离环境中。系统包括用于在第二安装范围内搜索对应于所请求原生资源的原生资源的装置。系统还获取位于第二隔离范围内的所请求原生资源的实例，并且以所获取的该原始资源的

实例响应对该原生资源的请求。

[0005] 在一个方面中，示出并且描述了用于关联安装范围以便于更新原生资源的实例的系统。该系统包括用于执行第一隔离环境中的第一应用和第二隔离环境中的第三应用的装置。该系统包括用于拦截对于原生资源的请求的装置，其中该原生资源可以被用在和位于第二隔离环境中的第三安装范围关联的第三应用中。在第三安装范围内对该原生资源的定位失败导致使用系统提供的装置来确定第三安装范围和代表第二应用映像的第二安装范围之间的关联。系统还包括用于获取位于第二安装范围内的所请求的原生资源的实例的装置。更新获取到的所请求原生资源的实例并且将其返回给第二安装范围。系统拦截用于获取原生资源的另一个请求。该请求由位于第一隔离环境中的第一安装范围所关联的第一应用所发送。在第一安装范围中对与所请求的原生资源相对应的原生资源的定位失败导致使用系统提供的装置做出关于第一安装范围和第二安装范围之间是否存在关联的决定。如果存在这样的关联，则系统获取位于第二安装范围内的更新后的原生资源的实例并且将所获取的实例返回给第一应用。

[0006] 在一个方面中，示出并且描述了用于便于在安装范围之间传递资源信息的隔离环境。该隔离环境包括代表第一应用的第一安装范围，其中第一应用根据第一安装范围内的可执行组件来执行。隔离环境中还包括代表第二应用的第二安装范围。第二安装范围还经由第二安装范围和第一安装范围的聚集与第一安装范围相关联。虚拟范围包括在隔离环境中。虚拟范围代表第一安装范围和第二安装范围的聚集。虚拟范围还提供对位于第一安装范围内的资源信息和位于第二安装范围内的资源信息的访问。

[0007] 在一个方面中，示出并且描述了用于从包含应用配置文件的用户范围获取用户设置信息的方法。所示方法包括初始化包括和第一应用相关联的第一安装范围的第一隔离环境。第一隔离环境包括用户范围，该用户范围包含还对应于第一应用的第一应用配置文件。从第一应用配置文件中获取和配置第一应用相关联的用户设置信息。第一应用根据所获取的用户设置信息在第一隔离环境中执行。该方法还包括初始化包括和第一应用相关联的第二安装范围的第二隔离环境。第二隔离环境还包括含有第一应用配置文件的用户范围。用户设置信息可从第一应用配置文件获取并且被用于配置第一应用。第一应用还根据所获取的用户设置信息在第二隔离环境中执行。

附图说明

[0008] 以下附图描写此处描述的方法和系统的示例性实施例。这些附图仅用于示例并且不限制此处描述的方法和系统。

[0009] 图 1A 是示出具有和服务器通信的客户机的远程访问网络化环境的实施例的框图；

[0010] 图 1B 和 1C 是示出用于实现此处描述的方法和系统的计算机器的实施例的框图；

[0011] 图 2A 是具有隔离环境的计算机器的是实施例的框图；

[0012] 图 2B 是具有隔离环境的多用户计算机器的实施例的框图；

[0013] 图 2C 是在隔离环境中启动执行应用程序的方法的实施例的流程图；

[0014] 图 3A 和 3B 是将原生资源的实例聚集到单个原生资源实例中的方法的实施例的流程图；

[0015] 图 4 是虚拟范围的实施例的框图；

[0016] 图 5 是获取原生资源的实例的方法的实施例的流程图；

[0017] 图 6A 是示出用于对访问计算机器上的原生资源进行虚拟化的方法的实施例的流程图；

[0018] 图 6B 是示出用于在执行模式时识别出对实例的替代的方法的实施例的流程图；

[0019] 图 6C 是示出用于在接收到打开原生资源的请求时识别文本 (literal) 资源的方法的实施例的流程图，该请求指示该资源是出于修改目的而被打开的；

[0020] 图 6D 示出用于处于安装模式时当接收到创建虚拟资源的请求时识别出文本资源的方法的实施例的流程图；

[0021] 图 7 是用于在安装范围内安装应用程序的方法的实施例的流程图；

[0022] 图 8 是描述用于打开所描述虚拟环境中的文件系统中的项的方法的实施例的流程图；

[0023] 图 9 是描述用于从所描述虚拟环境中的文件系统中删除项的方法的实施例的流程图；

[0024] 图 10 是描述用于在所描述虚拟环境中列举文件系统中的项的方法的实施例的流程图；

[0025] 图 11 是描述用于在所描述虚拟环境中创建文件系统中的项的方法的实施例的流程图；

[0026] 图 11A 是描述用于在所描述虚拟环境中创建文件系统中的文件项的方法的实施例的流程图；

[0027] 图 12 是描述用于在多于一个的安装范围上循环执行的方法的实施例的流程图；

[0028] 图 13 是描述用于在多于一个的安装范围上循环执行的方法的实施例的流程图；

[0029] 图 14 是描述用于打开所描述虚拟环境中的注册表键值的方法的实施例的流程图；

[0030] 图 15 是描述用于删除所描述虚拟环境中的注册表键值的方法的实施例的流程图；

[0031] 图 16 是描述用于列举所描述虚拟环境中的注册表数据库中的键值的子键值的方法的实施例的流程图；

[0032] 图 17 是描述用于在所描述虚拟环境中创建注册表键值的方法的实施例的流程图；

[0033] 图 18 是描述用于在多于一个的安装范围上循环执行的方法的实施例的流程图；

[0034] 图 19 是描述用于在多于一个的安装范围上循环执行的方法的实施例的流程图；

[0035] 图 20 是描述用于对访问命名对象进行虚拟化的方法的实施例的流程图；

[0036] 图 21A 是具有隔离环境的多用户计算机器的实施例的框图；

[0037] 图 21B 是虚拟范围的实施例的框图；

[0038] 图 21C 是将原生资源的实例聚集到单个原生资源实例中的方法的实施例的流程图。

具体实施方式

[0039] 提供应用程序隔离的计算环境

[0040] 图 1A 示出包括和服务器 106A-106N 通信的一个或者多个客户端机器 102A-102N 和安装在客户端机器 102A-102N 和服务器 106A-106N 之间的网络 104 的计算环境 101 的一个实施例。在一些实施例中，客户端机器 102A-102N 可以称之为单个客户端机器 102 或者单组客户端机器 102，而服务器 106 可以称之为单个服务器 106 或者单组服务器 106。另一个实施例包括和超过一个的服务器 106 通信的单个客户端机器 102，另一个实施例包括和超过一个的客户端机器 102 通信的单个服务器 106，又一个实施例包括和单个服务器 106 通信的单个客户端机器 102。

[0041] 在有些实施例中，计算环境中的客户端机器 102 可以称之为以下术语中的任一个：客户端机器 102、客户端、客户端计算机、客户端装置、客户计算装置、客户端节点、端点、端点节点，第二机器，或者表示和第一装置相连的第二装置的任一其它命名约定，其使得第二装置的操作部分依赖于第一装置执行的操作。在有些实施例中，服务器 106 可以称之为以下术语的中的任一个：服务器、服务器群组 (serverfarm)、宿主 (host) 计算装置，第一机器，或者表示和第二装置相连的第一装置的任一其它命名约定，其中第一装置可以至少部分管理第二装置的操作。

[0042] 在有些实施例中，客户端机器 102 可以执行、操作或以其他方式提供应用程序，所述应用程序可以是以下任意一个：软件，程序，可执行指令，web 浏览器，基于 web 的客户端，客户端 - 服务器应用程序，瘦 - 客户端的计算客户端，ActiveX 控件，Java 小程序，例如软 IP 电话的与 VoIP（基于网际协议的音频传输）通信相关的软件，用于流传送视频和 / 或音频的应用程序，便于实时数据通信的应用程序，HTTP 客户端，FTP 客户端，Oscar 客户端，Telnet 客户端，或可以在客户端机器 102 上执行的任一其它类型和 / 或形式的可执行指令。其他实施例还可以包括具有基于服务器或基于远程的应用程序以及代表客户端机器 102 在服务器 106 上执行的应用程序的计算环境 101。计算环境 101 的其他实施例包括配置成使用瘦 - 客户端或远程显示协议来显示输出图形数据到客户端机器 102 的服务器 106，其中所使用的协议可以是以下协议的任一：由位于 Ft. Lauderdale，Florida 的 Citrix Systems 公司出品的独立计算架构 (ICA) 协议或由位于 Redmond，Washington 的微软公司出品的远程桌面协议 (RDP)。

[0043] 在一些实施例中，计算环境 101 可以包括多于一个的服务器 106A-106N，其中该服务器 106A-106N：集合在一起作为单个服务器 106 实体、逻辑上集合在服务器群组 106 中；地理上分散并且逻辑上集合在服务器群组 106 中，互相位置邻近并且逻辑上集合在服务器群组 106 中。在一些实施例中，服务器群组 106 中的地理上分散的服务器 106A-106N 可以使用 WAN、MAN 或者 LAN 通信，其中不同的地理区域可以被表征为：不同的洲、洲的不同区域、不同国家、不同州、不同城市、不同校园、不同房间或者前述地理位置的任意组合。在一些实施例中，服务器群组 106 可以被管理为单一的实体，或者在其它实施例中可以包括多个服务器群组 106。计算环境 101 可以包括集合在单个服务器群组 106 中的多于一个的服务器 106A-106N，其中服务器群组 106 是非单一类型的，从而一个服务器 106A-106N 被配置为根据第一类型的操作系统平台（例如，由位于 Redmond，Washington 的微软公司出品的 WINDOWS NT）来进行操作，而一个或多个其它的服务器 106A-106N 被配置为根据第二类型的操作系统平台（例如，Unix 或 Linux）来进行操作；多于一个的服务器 106A-106N 被

配置为根据第一类型的操作系统平台（例如，WINDOWS NT）来进行操作，而另一个服务器106A-106N被配置为根据第二类型的操作系统平台（例如，Unix或Linux）来进行操作；或者多于一个的服务器106A-106N被配置为根据第一类型的操作系统平台（例如，WINDOWS NT）来进行操作，而多于一个的其它服务器106A-106N被配置为根据第二类型的操作系统平台（例如，Unix或Linux）来进行操作。

[0044] 在一些实施例中，计算环境101可以包括配置为提供以下服务器类型的任一的功能性的服务器106或者多于一个的服务器106：文件服务器，应用服务器，web服务器，代理服务器，设备，网络设备，网关，应用网关，网关服务器，虚拟服务器，部署服务器，SSL VPN服务器，防火墙，web服务器，应用服务器或者作为主应用服务器，配置成作为主动引导(active direction)的服务器106，配置成作为提供防火墙功能性、应用功能性或者负载平衡功能性的应用加速应用程序的服务器106，或者配置成用作服务器106的其它类型的计算机器。在一些实施例中，服务器106可以包括远程验证拨入用户服务，使得服务器106是RADIUS服务器。对于服务器106包括设备的计算环境101的实施例，服务器106可以是以下制造商的任一制造的设备：CitrixApplication Networking Group；Silver Peak Systems公司；RiverbedTechnology公司；F5 Networks公司；或Juniper Networks公司。一些实施例包括具有以下功能性的服务器106：第一服务器106A接收来自客户端机器102接收请求，将该请求转发给第二服务器106B，并且使用来自第二服务器106B的响应来响应客户端机器产生的该请求；获取客户端机器102可用的应用程序的列举和地址信息，该地址信息与用于寄载(host)该由应用程序的列举所识别出的应用程序的服务器106相关；使用web接口呈现对客户请求的响应，直接和客户端机器102通信来为客户端机器102提供对所识别应用程序的访问；接收通过在服务器106上执行所识别应用程序而产生的输出数据，诸如显示数据。

[0045] 服务器106可以被配置为执行以下应用程序中的任一个：提供瘦-客户端计算或远程显示演示的应用程序；Citrix Systems公司的CITRIX ACCESS SUITE的任一部分，例如METAFRAME或CITRIX演示服务器；由微软公司出品的任意一种微软Windows终端服务；或者Citrix Systems公司开发的ICA客户端。另一个实施例包括被配置为执行应用程序使得服务器可以用作应用服务器的服务器106，该应用服务器诸如以下应用服务器类型中的任一个：提供诸如微软公司出品的微软Exchange的电子邮件服务的电子邮件服务器，web或Internet服务器，桌面共享服务器，或协作服务器。其它的实施例包括的服务器106执行作为以下类型的运行在服务器上的应用程序中任一的应用程序：Citrix Online部门提供的GOTOMEETING，由位于SantaClara，California的WebEx公司提供的WEBEX，或由微软公司提供的微软Office Live Meeting。

[0046] 在一些实施例中，客户端机器102可以用作搜寻对服务器106提供的资源的访问的客户端节点，或者作为给其它客户端102A-102N提供对所驻留(host)资源的访问的服务器106。计算环境101的一个实施例包括提供主节点功能性的服务器106。客户端机器102和服务器106或者多个服务器106A-106N之间的通信可以经由以下方法的任一个来建立：客户端机器102和服务器群组106中的服务器106A-106N之间的直接通信；客户端机器102使用程序邻里(programneighborhood)应用程序和服务器群组106中的服务器106a-106n通信；或者客户端机器102使用网络104和服务器群组106中的服务器106A-106N通信。

计算环境 101 的一个实施例包括客户端机器 102，该客户端机器 102 使用网络 104 来请求由服务器群组 106 中的服务器 106A-106N 运行 (host) 的应用程序执行，并且使用网络 104 从服务器 106A-106N 接收表示应用程序执行的图形显示输出。在其他实施例中，主节点提供识别和提供与寄载所请求应用程序的服务器 106 相关联的地址信息所需的功能性。其他实施例包括的主节点可以是以下的任一个 ：服务器群组 106 中的服务器 106A-106N ；连接到服务器群组 106 但不包括在服务器群组 106 中的远程计算机器 ；连接到客户端 102 但不包括在一组客户端机器 102 中的远程计算机器 ；或者客户端机器 102。

[0047] 客户端机器 102 和服务器 106 之间的网络 104 是通过其在客户端机器 102 和服务器 106 之间传送数据的连接。尽管图 1A 示出将客户端机器 102 连接到服务器 106 的网络 104，但是其他实施例包括的计算环境 101 中客户端机器 102 与服务器 106 安装在同一网络中。其他实施例可以包括具有能够是以下任一个的网络的计算环境 101 ：局域网 (LAN)，城域网 (MAN)，广域网 (WAN)，包括位于客户端机器 102 和服务器 106 之间的多个子网 104’的主网络 104，具有专用子网 104’ 的主公用网 104，具有公用子网 104’ 的主专用网 104，或者具有专用子网 104’的主专用网 104。其它的实施例包括可以是以下网络类型的任一个的网络 104 ：点到点网络、广播网、远程通信网、数据通信网、计算机网络、ATM（异步传送模式）网络、SONET（同步光网）网络、SDH（同步数字系列）网络、无线网络、有线网络，网络 104 包括无线链路，该无线链路可以是红外信道或卫星频带，或者能够将数据从客户端机器 102 传送到服务器 106 并且反向传输来实现此处描述的方法和系统的任意其他网络类型。网络拓扑在不同实施例中可以不同，可能的网络拓扑包括 ：总线形网路拓扑、星形网络拓扑、环形网络拓扑、基于转发器的网络拓扑、分层星形网络拓扑，或者能够将数据从客户端机器 102 传送到服务器 106 并且反向传输来实现此处描述的方法和系统的任意其他网络拓扑。另外的实施例可以包括移动电话网络的网络 104，其使用协议在移动装置间通信，其中该协议可以是以下任一个 ：AMPS、TDMA、CDMA、GSM、GPRS UMTS，或者能够在移动装置间传输数据来实现此处描述的方法和系统的任意其他协议。

[0048] 图 1B 所示是可以用作图 1A 中所示的客户端机器 102 和服务器 106 的计算装置 100 的实施例。计算装置 100 中包括和以下部件通信的系统总线 150 ：中央处理单元 121，主存储器 122，存储装置 128，输入 / 输出 (I/O) 控制器 123，显示装置 124A-124N，安装装置 116 和网络接口 118。在一个实施例中，存储装置 128 包括 ：操作系统、软件程序和客户机代理 120。在一些实施例中，I/O 控制器 123 还连接到键盘 126 和指示装置 127。其他实施例可以包括连接到多于一个的输入 / 输出装置 130A-130N 的 I/O 控制器 123。

[0049] 图 1C 示出可以用作客户端机器 102 和服务器 106 的计算装置 100 的一个实施例。计算装置 100 中包括和以下部件通信的系统总线 150 ：桥接器 170、第一 I/O 装置 130A。在另一个实施例中，桥接器 170 还和中央处理单元 121 相通信，其中中央处理单元 121 还和第二 I/O 装置 130B、主存储器 122 以及高速缓冲存储器 140 相通信。中央处理单元 121 中包括 I/O 端口、存储器端口 103 和主处理器。

[0050] 计算机器 100 的实施例可以包括中央处理单元 121，其特征在于以下部件配置的任一个 ：响应并处理从主存储器单元 122 获取的指令的逻辑电路 ；微处理器单元，诸如 ：由 Intel 公司出品的产品 ；由 Motorola 公司出品的产品 ；由位于 Santa Clara，California 的 Transmeta 公司出品的产品 ；由国际商业机器公司出品的 RS/6000 处理器 ；或者由 Advanced

Micro Devices 公司出品的处理器；或者是能够执行如此处所描述的系统和方法的任一其它组合的逻辑电路。中央处理单元 122 的其他实施例包括以下的任一组合：微处理器、微控制器、具有单个处理核的中央处理单元、具有两个处理核的中央处理单元，或者具有多于一个的处理核的中央处理单元。

[0051] 计算机器 100 的一个实施例包括通过也被称为后端总线的次级总线来与高速缓冲存储器 140 通信的中央处理单元 121，而计算机器 100 的另一个实施例包括经由系统总线 150 和高速缓冲存储器通信的中央处理单元 121。在一些实施例中，中央处理单元可以使用本地系统总线 150 与多于一种类型的 I/O 装置 130A-130N 进行通信。在一些实施例中，本地系统总线 150 可以是以下类型总线的任一个：VESA VL 总线、ISA 总线、EISA 总线、微通道架构 (MCA) 总线、PCI 总线、PCI-X 总线、PCI-Express 总线或 NuBus。计算机器 100 的其他实施例包括作为视频显示器 124 的 I/O 装置 130A-130N，中央处理单元 121 可以使用高级图形端口 (AGP) 与视频显示器 124 进行通信。计算机器 100 的其他形式包括经由以下连接的任一个连接到 I/O 装置 130A-130N 的处理器 121：HyperTransport、快速 I/O 或 InfiniBand。计算机器 100 的其他实施例包括通信连接，其中处理器 121 使用本地互连总线与 I/O 装置 130A 进行通信，并且使用直接连接与第二 I/O 装置 130B 进行通信。

[0052] 计算装置 100 的一些实施例中包括主存储器单元 122 和高速缓冲存储器 140 的每一个。在一些实施例中，高速缓冲存储器 140 可以是以下类型存储器任一个：SRAM、BSRAM 或 EDRAM。其它实施例包括高速缓冲存储器 140 和可以是以下类型存储器任一的主存储器单元 122：静态随机存取存储器 (SRAM)、突发式 SRAM 或同步突发式 SRAM(BSRAM)、动态随机存取存储器 (DRAM)、快速页面模式 DRAM(FPM DRAM)、增强型 DRAM(EDRAM)、扩展数据输出 RAM(EDO RAM)、扩展数据输出 DRAM(EDO DRAM)、突发式扩展数据输出 DRAM(BEDO DRAM)、增强型 DRAM(EDRAM)、同步 DRAM(SDRAM)、JEDEC SRAM、PC100 SDRAM、双数据速率 SDRAM(DDR SDRAM)、增强型 SDRAM(ESDRAM)、同步链接 DRAM(SLDRAM)、直接 Rambus DRAM(DRDRAM)、或铁电 RAM(FRAM)，或者能够执行此处描述的系统和方法的任意其它类型的存储装置。在一些实施例中，主存储器单元 122 和 / 或高速缓冲存储器 140 可以包括能够保存数据并且允许中央处理单元 121 直接访问任一存储位置的一个或者多个存储器装置。另外的实施例包括可以经以下任一访问主存储器 122 的中央处理单元 121：系统总线 150、存储器端口 103，或者允许处理器 121 访问存储器 122 的任意其它连接、总线或者端口。

[0053] 计算装置 100 的一个实施例提供对任意以下的安装装置 116 的支持：例如用于容纳像 3.5 英寸、5.25 英寸磁盘或 ZIP 磁盘这样的软盘的软盘驱动器、CD-ROM 驱动器、CD-R/RW 驱动器、DVD-ROM 驱动器、多种格式的磁带驱动器、USB 装置、可引导介质、可引导 CD、诸如KNOPPIX®的可用作 GNU/Linux 分发的可引导 CD、硬盘驱动器或适于安装应用程序或者软件的任意其它装置。在一些实施例中，应用程序包括客户端代理 120 或者客户端代理 120 的任一部分。计算装置 100 还可以包括存储装置 128，其例如是一个或多个硬盘驱动器或一个或多个独立磁盘的冗余阵列；其中存储装置被配置用于保存操作系统、软件、程序应用、或者客户端代理 120 的至少一部分。计算装置 100 的另一个实施例包括用作存储装置 128 的安装装置 116。

[0054] 进一步地，计算装置 100 可以包括通过多种连接联接到局域网 (LAN)、广域网 (WAN) 或者因特网的网络接口 118，所述多种连接包括但不限于标准电话线、LAN 或 WAN 链路

(例如,802.11、T1、T3、56kb、X.25、SNA、DECNET)、宽带连接(例如,ISDN、帧中继、ATM、吉比特以太网、SONET 上以太网)、无线连接或上述任意或所有连接的一些组合。连接可以使用多种通信协议来建立(这些协议例如 TCP/IP、IPX、SPX、NetBIOS、以太网、ARCNET、SONET、SDH、光纤分布数据接口 (FDDI)、RS232、RS485、IEEE 802.11、IEEE802.11a、IEEE 802.11b、IEEE 802.11g、CDMA、GSM、WiMax 和直接异步连接。计算装置 100 的一个版本包括可以经由任一类型和 / 或形式的网关或者隧道协议和其他计算装置 100' 通信的网络接口 118,这些网关或者隧道协议诸如安全套接字层 (SSL) 或者传输层安全 (TLS)、或者 Citrix System 公司出品的 Citrix 网关协议。网络接口 118 的形式可以包括以下任一个 :内置网络适配器、网络接口卡、PCMCIA 网卡、卡总线网络适配器、无线网络适配器、USB 网络适配器、调制解调器或适于将计算装置 100 连接到能够传达并执行此处所描述的方法和系统的网络的任意其它装置。

[0055] 计算装置 100 的实施例包括以下 I/O 装置 130A-130N 的任一个 :键盘 126、指示装置 127、鼠标、触控板、光笔、轨迹球、麦克风、绘画板、视频显示器、扬声器、喷墨打印机、激光打印机和染料升华打印机,或者可以执行此处描述的方法和系统的任一其它输入 / 输出装置。在一些实施例中,I/O 控制器 123 可以连接到多个 I/O 装置 103A-103N 以控制一个或者多个 I/O 装置。I/O 装置 130A-130N 的一些实施例可以被配置为提供存储或安装介质 116,而其他可以提供用于接收通用串行总线 (USB) 存储装置 USB 接口,诸如 TwintechIndustry 公司出品的 USB 闪速驱动系列。I/O 装置 130 的其他实施例可以是系统总线 150 和外部通信总线之间的桥接部件,外部通信总线例如 USB 总线、Apple Desktop 总线、RS-232 串行连接、SCSI 总线、FireWire 总线、FireWire 800 总线、以太网总线、AppleTalk 总线、吉比特以太网总线、异步传送模式总线、HIPPI 总线、超 HIPPI 总线、SerialPlus 总线、SCI/LAMP 总线、FibreChannel 总线或串行附加小型计算机系统接口总线。

[0056] 在一些实施例中,计算装置 100 可以连接到多个显示装置 124A-124N,在其他实施例中,计算装置 100 可以连接到单个显示装置 124,而仍然在其他实施例中,计算装置 100 可以连接到相同类型或者形式显示的显示装置 124A-124N,或者不同类型或者形式显示的显示装置 124A-124N。显示装置 124A-124N 的实施例可以通过以下支持并启用 :一个或者多个 I/O 装置 130A-130N、I/O 控制器 123、I/O 装置 130A-130N 和 I/O 控制器 123 的组合,可以支持显示装置 124A-124N 的硬件和软件的任意组合,任意类型和 / 或形式的视频适配器、视频卡、驱动器和 / 或库,以联接、通信、连接或以其他方式使用显示装置 124a-124n。在一些实施例中,计算装置 100 可以被配置为使用一个或者多个显示装置 124A-124N,这些配置包括 :使用多个连接器联接到多个显示装置 124a-124n,具有多个视频适配器,每个视频适配器连接到一个或多个显示装置 124A-124N,具有被配置用于支持多个显示器 124A-124N 的操作系统。使用包括在计算装置 100 中的电路和软件来连接到并且使用多个显示装置 124A-124N,并且执行主计算装置 100 和多个次级计算装置上的软件以启用主计算装置 100 来使用次级计算装置的显示器作为主计算装置 100 的显示装置 124A-124N。计算装置 100 的其他实施例可以包括由多个次级计算装置提供并且经由网络连接到主计算装置 100 的多个显示装置 124A-124N。

[0057] 在计算机器 100 的一些实施例中,可以包括操作系统来控制任务的调度和对系统资源的访问。计算装置 100 的实施例能够运行以下任一操作系统 :例如 WINDOWS 3.x、

WINDOWS 95、WINDOWS 98、WINDOWS 2000、WINDOWS NT 3.51、WINDOWS NT 4.0、WINDOWS CE、WINDOWS XP 和 WINDOWS VISTA 的任意一种版本的微软 Windows 操作系统、不同版本的 Unix 和 Linux 操作系统、苹果计算机公司出品的任意版本的 MAC OS、国际商业机器公司出品的 OS/2、任意的嵌入式操作系统、任意的实时操作系统、任意的开源操作系统、任意的专用操作系统、用于移动计算装置的任意操作系统、或者能够运行在计算装置上运行并执行此处所描述的操作的任意其它操作系统。计算机器 100 的一个实施例在其上安装多个操作系统。

[0058] 计算装置 100 能以以下形式因子实现 :计算工作站、台式计算机、膝上型或笔记本计算机、服务器、便携式计算机、移动电话、便携电信装置、媒体播放装置、游戏系统、移动计算装置、苹果计算机公司出品的 IPOD 家族的装置、Sony 公司出品的 PLAYSTATION 家族的任一个、Nintendo 公司出品的 NINTENDO 家族的任一个,微软公司出品的 XBOX 家族的任一个,或者能够通信并具有执行此处所描述的方法和系统的足够的处理器能力和存储容量的其它类型和 / 或形式的计算、电信或媒体装置。在其它实施例中,计算装置 100 可以是诸如以下移动装置中任一的移动装置 :支持 JAVA 的蜂窝电话或者个人数字助理 (PDA),例如 Motorola 公司出品的 i55sr、i58sr、i85s、i88s、i90c、i95cl 或者 iml100,Kyocera 出品的 6035 或者 7135,或者三星电子公司出品的 i300 或者 i330,由 Palm 公司出品的 TREO 180、270、600、650、680、700p、700w 或者 750 智能电话,具有不同的处理器、操作系统和与所述装置一致的输入装置的计算装置,或者可以执行此处描述的方法和系统的任意其它移动计算装置。计算环境 101 的其他实施例还包括可以是以下任一个的移动计算装置 100 :Research In Motion 有限公司出品的黑莓系列或者其它手持装置系列,苹果计算机公司出品的 iPhone,任一手持或者智能电话,Pocket PC,Pocket PC Phone,或者支持 Microsoft Windows Mobile Software(微软视窗移动软件)的其它手持移动装置。

[0059] 如此处所参考的,范围可以是安装范围、系统范围、用户范围、应用程序专用的用户范围、隔离范围或者其他范围的任一个。在一个实施例中,系统范围是指位于操作系统内的系统根目录中的范围。在此实施例中,系统范围可以包括和计算机器 200 相关联的原生资源。在一个实施例中,安装范围是指包括和应用程序相关联的原生资源的范围。其它的实施例所包括的安装范围可以包括以下任一个 :和安装范围相关联的应用程序所使用的原生资源的实例,和安装范围相关联的应用程序的实例,和安装范围相关联的应用程序所使用的一组原生资源,和安装范围内的一个或者多个应用程序、应用程序关联或者应用程序实例相关联的超过一组的原生资源或者原生资源实例,或者一组应用程序专用设置和配置。在一些实施例中,用户范围可以是包括对应于用户专用的设置和配置的原生资源的实例的范围。应用程序专用的用户范围包括一类应用程序专用的原生资源,其中通过用户输入确定该设置。应用程序专用的用户范围中的原生资源的实例的一个例子包括对于 Microsoft Word 应用程序的用户设置。隔离范围可以是包括在隔离环境中任一范围,或者有助于隔离环境中应用程序的任一范围。隔离范围的例子包括安装范围、用户范围和应用程序专用的范围。在特定实施例中,系统范围还可以是隔离范围。在一些实施例中,范围是一个或者多个应用程序所使用来执行的包括原生资源、设置和配置的组。其他实施例包括具有原生资源的实例的范围,而另一些其他实施例还包括具有原生资源的实例和一个或者多个和该范围关联的应用程序的实例的范围。在一些实施例中,范围还可以包括子范围,并且范围还可

以包括超过一个的单个范围。

[0060] 图 2A 所示是计算环境中的计算机器 200 的一个实施例，其中计算机器 200 以降低的应用程序兼容性和应用程序社交性进行操作。安装在计算机器 200 上的是操作系统 210，其进一步和一个或者多个应用程序 203 通信，应用程序 203 包括“应用程序 1”202 和“应用程序 2”204。操作系统 210 中是系统根目录 (root)216、安装根目录 214 和用户根目录 212。安装根目录 214 和用户根目录 212 一起驻留在也位于操作系统 210 中的隔离环境 260 中。安装根目录 214 中有各个安装范围 230，诸如：“安装范围 1”224、“安装范围 2”226 和“安装范围 3”228。

[0061] 继续参考图 2A 并且更详细地，实施例可以包括提供隔离环境 260 以减轻应用程序兼容性和应用程序社交性的计算机器 200。当超过一个的应用程序或者类似应用程序的实例由一个或者多个用户在计算机器 200 上执行时，则产生应用程序兼容性问题。这些应用程序通常利用保存在计算机器 200 上的系统根目录 216 中的类似原生资源。当一个应用程序掌控或者以其他方式改变原生资源，而第二应用程序或者第一应用程序的第二实例利用刚刚改变的同一原生资源时，则会危及第二应用程序或者该应用程序实例的执行。该第二应用程序 / 应用程序实例的正确执行需要该原生资源具有特定的形式或者具有特定的值。由于当超过一个的应用程序或者应用程序实例在计算机器 200 上执行时每一个应用程序或者应用程序实例的正确执行都需要访问特定的原生资源，因而应用程序社交性是类似的概念。如果这些所请求的原生资源在应用程序和应用程序实例间是共有的，则会产生关于应用程序是否能够访问具有正确值和形式的原生资源的问题。

[0062] 在一个实施例中，计算机器 200 是配置为经由网络 104 和服务器 106 通信的客户端计算机器 102。另一个实施例所包含的计算机器 200 是可以经由网络和客户端机器 102 通信的服务器 106。而又一些实施例所包含的计算机器 200 可以是计算机器的上述配置的任一个。

[0063] 在一个实施例中，操作系统 210 是安装在计算机器 200 上用来控制包括在计算机器 200 中的各部件的操作的程序应用。在其他实施例中，操作系统 210 可以是上述操作系统的任一组合。

[0064] 在一个实施例中，计算机器 200 上的应用程序 203 和操作系统 210 相通信。在其他实施例中，应用程序 203 可以直接和包括在计算机器 200 中的部件相通信，或者应用程序 203 不和操作系统 210 相通信。在一个实施例中，单个应用程序 202 可以和操作系统 210 相通信，或者多个应用程序 203 可以和操作系统 210 相通信。在其他实施例中，应用程序 203 还可以是任一程序、例程、命令集或者其它软件应用。

[0065] 在一个实施例中，包括系统根目录 216。系统根目录 216 提供专用于计算机器 200 的原生资源集。在另一个实施例中，具有原生资源的系统根目录 216 基本类似于当仅将操作系统 210 安装在计算机器 200 上时呈现在计算机器 200 上的那些原生资源。

[0066] 一个实施例包括将单个的安装范围 230 集合在一起的安装根目录 214。在一个实施例中，安装根目录 214 根据在每个安装范围 230 中隔离的应用程序的需求而将关联的单个安装范围 230 集合在一起。在另一个实施例中，安装根目录 214 根据以下任一将单个安装范围集合在一起：应用程序策略，应用程序设置，安装根目录 214 的设置，安装范围 230 设置，或者可以用来识别集合在单个安装根目录 214 中的安装范围 230 的任一其它标准。在

一个实施例中，安装根目录 214 提供应用程序隔离，其中所提供的该隔离减轻了和应用程序社交性和兼容性相关的问题。又一个实施例中，安装根目录 214 具有主要基础安装范围和配置来用作主要基础安装范围的子范围的附加安装范围。在此实施例中，安装范围包括对主要基础安装范围的修改，其中该修改可以是应用程序的补丁级中的改变，或者是对附加的应用程序特征的安装或者移除。

[0067] 许多实施例包括安装根目录 214 中的安装范围 230。每个安装范围 230 包括和特定应用程序相关联的原生资源的实例。例如，第一应用程序可以安装在第一安装范围 224 中。该第一安装范围 224 中可以包括专用于安装在第一安装范围 224 中的第一应用程序的原生资源的实例。在一个实施例中，安装范围 230 的每一个包括专用于应用程序 203 的原生资源的视图。例如，专用于第一应用程序 202 的原生资源可以表示在第一安装范围 224 中，而专用于第二应用程序 204 的原生资源可以表示在第二安装范围 226 中。其他实施例包括和第一安装范围 224 和第二安装范围 226 相关联的第三安装范围 228，使得第一应用程序 202 和第二应用程序 204 二者可以访问和操控包括在第三安装范围 228 中的原生资源的视图。其他实施例包括集合在一起的应用程序 203，使得单个安装范围 230 可以为每组应用程序 203 提供原生资源的特定视图。在此实施例中，发生冲突的应用程序 203 可以被分开到不同的组中，以便提供应用程序的兼容性和社交性。在其他实施例中，管理员可以配置属于一个应用程序集合的应用程序。在又一些实施例中，“通透”安装范围可以被限定为其精确对应于包括在系统根目录 216 中的原生资源的实例。换句话说，在通透安装范围中执行的应用程序直接操作包括在系统根目录 216 中的那些原生资源。在一些实施例中，安装范围可以被实现为对于执行的应用程序的实例而言是可见的，从而执行的应用程序可以配置该可见的安装范围。在其他实施例中，可见的安装范围集包括对于执行的应用程序的所有实例而言共有的原生资源实例集，其与执行的应用程序所代表的用户无关。其他实施例包括具有原生资源实例的可变集的可见安装范围集，其根据执行该应用程序的用户而改变。一个实施例包括安装范围 230，其中附加的安装范围 230 可以基于用户定义的参数在安装根目录 214 中执行。这些参数可以包括关于哪个安装范围应该执行的指令，或者用于实现执行一些安装范围 230 而不执行其他安装范围的操作指令。在一些实施例中，当安装范围不再需要时丢弃该安装范围，而在其它实施例中，安装范围 230 可以合并到单个安装范围 230 中，该安装范围 230 包括表示包括在每个单独安装范围中的单独的原生资源实例的原生资源的实例。

[0068] 仍旧参考图 2A，在一个实施例中，包括一个用户根目录 212，其保存和提供表示用户定义的设置和配置信息的原生资源的实例。在一些实施例中，用户根目录 212 可以包括应用程序 203 根据用户定义的设置进行初始化所使用的原生资源的实例。一个实施例中的隔离环境 260 不包括用户根目录 212，而另一个实施例包括的安装根目录 214 所包括的范围大体类似于用户根目录 212 中包括的范围。其他实施例包括的用户根目录 212 所具有的原生资源的实例将包括在安装根目录 214 中的应用程序专用的设置和配置与用户定义的设置和配置组合在一起。

[0069] 在一个实施例中，包括还进一步包含用户根目录 212 和安装根目录 214 的隔离环境 260。其他实施例可以包括具有任一以下特征的隔离环境 260 ：具有超过一个的用户根目录 212 的隔离环境 260，具有超过一个的安装根目录 214 的隔离环境 260，包括用户根目录

212、安装根目录 214 和系统根目录 216 的隔离环境 260，包括安装根目录 214 和系统根目录 216 的隔离环境 260，包括用户根目录 212 和系统根目录 216 的隔离环境 260，包括用户根目录 212 的隔离环境 260，包括系统根目录 216 的隔离环境 260，包括安装根目录 214 的隔离环境 260，包括附加范围或者根目录（未示出）的隔离环境 260，或者包括根据用户配置文件或者应用程序配置文件集合在一起的多个安装范围和用户根目录的隔离环境 260。在一个实施例中，隔离环境 260 通过隔离专用于应用程序的原生资源和专用于执行该应用程序的用户的原生资源而提供对执行的应用程序的隔离。对专用于应用程序的原生资源的隔离包括将包括在安装根目录 214 和用户根目录 212 中的原生资源的那些实例提供给执行的应用程序，而不是允许该应用程序访问和掌控系统根目录 216 中的原生资源。原生资源在应用程序之间可以不同，从而给执行的应用程序提供其自己独有的原生资源集可以减轻应用程序的非兼容性和非社交性。

[0070] 图 2B 中所示的实施例被配置为提供对多于一个的用户 238、240、242 的访问并且具有降低的应用程序兼容性和应用程序社交性问题。所示计算机器 200 具有：安装在其上的操作系统 210，指示用户动作的用户会话 238、240、242，和在用户会话 238、240、242 中执行的应用程序 202、204、206、208。操作系统 210 具有系统根目录 216、安装根目录 214 和用户根目录 211。系统根目录 216 中包括的系统范围 256 还包括诸如以下的原生资源 258：文件系统 218、注册表 220 和对象 222。安装根目录 214 包括超过一个的安装范围 244、246、252、248、250、254，其中每一个安装范围连接到另一个范围。每一个安装范围中包括原生资源的实例 258’、258”、258”’、258””、258””’、258””””。用户根目录 212 包括超过一个的用户范围 232、234、236，其中每一个范围和用户会话 238、240、242 相关联，并且每一个范围包括原生资源的实例。操作系统 210 中包括的隔离环境 260 将用户根目录 212 和安装根目录 214 集合在一起。

[0071] 继续参考图 2B 并且更详细地，在一个实施例中，计算机器 200 可以是上述计算机器 100 的任一组合。在另一个实施例中，其上安装的操作系统 210 可以是上述操作系统的任一组合。另一些其他实施例可以包括应用程序 202、204、206、208，其中应用程序可以是以下任一个：安装在计算机器 200 上的软件程序，计算机器 200 远程访问的应用程序，发送到操作系统 210 的软件命令集，软件例程，操作系统，或者可以在计算机器 200 上执行的任一其它应用程序。在一个实施例中，计算机器 200 可以用作客户端机器 102 或者服务器 106。

[0072] 实施例可以包括操作系统 210，该操作系统 210 还可以包括系统根目录 216。在一个实施例中，系统根目录 216 还包括原生资源 258 集合在其中的系统范围 256。其他实施例中的系统根目录 216 所具有的原生资源在系统根目录 216 中集合在一起。另一些其他实施例所包括的操作系统 210 不具有系统根目录 216，而具有位于操作系统 210 中的一组系统原生资源 258。原生资源 258 可以不集合在一起，而是分散遍及系统根目录 216、系统范围 256 或者操作系统 210 中的任一个。一个实施例包括具有多个系统范围（未示）的系统根目录 216，其中每一个系统范围可以具有独有的原生资源实例集。

[0073] 在一个实施例中，一组原生资源 258 和原生资源的实例可以是以下资源类型的任一个：文件系统 218，注册表 220，系统对象 222，窗名称，或者是对于正确执行应用程序有用的任一其它原生资源。当应用程序需要特定资源来正确执行时，由应用程序 202、204、206、208 请求原生资源 258。例如，应用程序可以请求保存在注册表键值中的特定设置，以执行

该应用程序的一部分。该部分应用程序的执行规定该应用程序从注册表 220 中请求获得包含所需设置的键值。一些实施例包括原生资源 258 的组，其中该组可以包含多于一个的原生资源，或者具有不同原生资源类型的多组原生资源 258。其他实施例可以包含以下组合的任一个：具有原生资源 258 的系统范围，具有原生资源 258 的实例的用户范围，和具有原生资源 258 的实例的安装范围；具有原生资源 258 的系统范围，和具有原生资源 258 实例的安装范围；具有原生资源 258 的系统范围，和具有原生资源 258 的实例的用户范围；具有原生资源 258 的实例的系统范围，具有原生资源 258 的实例的用户范围，具有原生资源 258 的实例的安装范围；或者隔离范围和原生资源 258 的任一组合。

[0074] 在许多实施例中，安装根目录 214 提供原生资源 258 的修改视图。该修改视图将包括在安装根目录 214 中的各个单个的安装范围 230 聚集在一起，来向发出请求的应用程序提供专用于该应用程序的原生资源实例。在一些实施例中，安装根目录 214 可以和用户根目录 212 以及系统根目录 216 直接通信，而在其他实施例中，安装根目录 214 可以经由安装根目录 214 中的安装范围 230 和其他根目录通信。其他实施例可以包括任一上述的安装根目录 214 的配置。

[0075] 在一个实施例中，安装范围 230 包括在安装根目录 214 中并且还被配置为和包括在用户根目录 212 中的用户范围相通信。其他实施例包括的安装范围可以和一个系统范围 256 相通信、和多于一个的系统范围 256 相通信或者不和其他范围相通信。在一个实施例中，安装范围 230 可以和应用程序 203、另一个安装范围 230 或者应用程序 203 和安装范围 230 这二者建立关联。在一个实施例中，安装范围 230 被配置为向发出请求的应用程序提供专门用于该发出请求的应用程序的原生资源 258 的实例。在此实施例中，安装范围 230 配置为表示单个应用程序或者在其他实施例中表示多个应用程序，从而所表示的安装范围包含相应的应用程序所需的原生资源。一个实施例包括由第一安装范围 244 表示并和其关联的第一应用程序 202，使得第一应用程序 202 可以访问和掌控包括在第一安装范围 244 中的原生资源 258 的视图；该实施例还包括由第二安装范围 246 表示并和其关联的第二应用程序 204，使得第二应用程序 204 可以访问和掌控包括在第二安装范围 246 中的原生资源 258”的视图。在此实施例中，第一安装范围 244 和第二安装范围 246 还可与第三安装范围 252 相关联，使得第一安装范围 244 和第二安装范围 246 的其中一个或者二者可以访问和掌控包括在第三安装范围 252 中的原生资源 258”’的视图。另一个实施例包括由第四安装范围 248 表示并和其关联的第三应用程序，使得第三应用程序 206 可以访问和掌控包括在第四安装范围 248 中的原生资源 258 的视图。又一个示例性实施例包括由第五安装范围 250 表示并和其关联的第四应用程序 208，使得第四应用程序 208 可以访问和掌控包括在第五安装范围 250 中的原生资源 258 的视图。在此实施例中，第五安装范围 250 还可以和第六安装范围 254 相关联，使得第四应用程序 208 可以访问和掌控包括在第六安装范围 254 中的原生资源 258 的视图。

[0076] 在一个实施例中，包括用户根目录 212。用户根目录 212 还包括用户范围 232、234、236，其中每个用户范围和安装根目录 214 中的安装范围相通信。计算机器 200 的实施例包括具有用户配置文件的用户根目录 212，其中用户配置文件保存可以由在用户会话中执行的应用程序所访问的用户专用的设置和配置信息。其他实施例包括具有多于一个的用户根目录 212 的计算机器 200，其中所包括的每一个用户根目录 212 包含用户专用的设置和配

置信息并且其中每一个用户根目录 212 对应于一个特定用户。其他实施例不包括用户根目录 212，而是包括安装根目录 214，该安装根目录 214 包括用于保存对应于用户指定设置和配置的原生资源的实例的安装范围。一个实施例包括具有多个用户范围的用户根目录 212。在此实施例中，一个用户范围（未示）可以专用于保存专用于特定应用程序的用户指定设置和配置。例如，用户范围可以配置为保存对应于用于 Microsoft Word® 的用户指定的设置的原生资源的实例。在此例中，每当 Microsoft Word® 应用程序在特定用户会话中执行时，用户所指定的 Microsoft Word® 配置和设置可以用于 Microsoft Word® 应用程序。

[0077] 在一个实施例中，用户范围包括在用户根目录 212 中。在一个实施例中，用户范围 232、234、236 的每一个对应于用户会话 238、240、242。用户范围 232、234、236 包括原生资源 258 的实例，其中每一实例表示专用于特定用户会话 238、240、242 的设置和配置。其他实施例包括含有子范围的用户范围 232、234、236。在此实施例中，呈现给在该范围内执行的应用程序的原生资源视图是对包括在每一个用户范围子范围中的修改的聚集 (aggregate)。在一个实施例中，子范围彼此互相层叠，并且在聚集的视图中较高子范围中对资源的修改覆盖较低层中对同一资源的修改。其他实施例包括的用户范围 232、234、236 包含专用于用户集的原生资源实例。其它的实施例包括用户范围 232、234、236，其中该原生资源实例由系统管理员限定，或者由在操作系统 210 中限定的一组预定用户来限定。另一个实施例可以包括用户范围，其中一些用户范围包括专用于特定用户会话的原生资源的实例。在此实施例中，用户范围可以保持并且维持对于未来登录会话的可用性，而其他实施例包括在用户会话结束时被破坏的用户范围。在一些实施例中，包括在用户范围中的原生资源可以由在相应用户会话中执行的应用程序来改变，而在其他实施例中，原生资源实例可以由用户输入改变。在计算机器 200 中使用用户范围的一个例子包括具有在第一用户会话 238 中执行的第一应用程序 202 的用户范围，具有概述用户专用的设置、配置以及第一应用程序 202 专用的组成的第一用户配置文件。在此例中，第一用户范围 232 根据第一用户配置文件修改原生资源的视图，并且向该第一应用程序 202 提供在第一用户会话 238 期间对这些用户指定的原生资源的访问。在计算机器 200 中使用用户范围的另一个例子包括具有在第三用户会话 242 中执行的第三应用程序 206 和第四应用程序 208 的用户范围，具有概述用户专用的设置、配置以及第三应用程序 206 专用的组成的第三用户配置文件。在此例中，第三用户范围 236 根据第三用户配置文件修改原生资源的视图并且向第三应用程序 206 和第四应用程序 208 提供在第三用户会话 242 期间对这些用户指定的原生资源的访问。另一种方式，通过将用户隔离范围 232、234、236 提供的用户专用的、修改后的视图修改层叠在安装根目录 214 提供的应用程序专用的、修改后的视图之上，而该应用程序专用的、修改后的视图又层叠在系统根目录 216 提供的原生资源的全系统视图之上，用户根目录 212 改变用于每一单独用户的原生资源的视图。

[0078] 仍旧参考图 2B，在一个实施例中，用户会话 238、240、242 和应用程序 202、204、206、208 包括在计算机器 200 中。用户会话 238、240、242 对应于用户，使得当用户利用计算机器 200 时，建立用户会话并且将用户产生的任一用户输入和用户输出限制到该用户会话。在此实施例中，当用户执行应用程序时，在用户会话中执行该应用程序。在计算机器 200 的一些实施例中，多个用户可以访问多个用户会话。仍旧其他实施例可以提供同时由超过一个的用户会话访问的应用程序 202、204、206、208。

[0079] 计算机器 200 的一个实施例包括隔离环境 260。隔离环境 260 包括用户根目录 212 和安装根目录 214。其他实施例可以包括诸如上述隔离环境的任一个的隔离环境 260。在计算机器 200 的一个实施例中，尽管所做描述是关于支持不同用户并行执行应用程序的多用户计算机，但是隔离环境 260 还可以被用在单用户计算机上来解决由不同用户在同一计算机系统上顺序执行应用程序所导致的应用程序兼容性和社交性问题，以及同一用户安装和执行不兼容程序所导致的那些问题。在一些实施例中，安装范围或者安装子范围可以和单独的线程而不是整个进程相关联，允许基于每线程来执行隔离。在一些实施例中，每线程隔离可以用于服务和 COM+ 服务器。

[0080] 在一个实施例中，安装范围中隔离应用程序要求应用程序安装在特定安装范围中（以下详述），并且该应用程序在该特定安装范围中开始。在应用程序安装到安装范围的实施例中，所安装的应用程序继续和它们各自的安装范围相关联，直到改变了和安装范围相关联的设置。其它的实施例包括在特定安装范围或者超过一个的安装范围中启动的应用程序，并且结果变成和一个或者多个安装范围相关联。在许多实施例中，当所关联的应用程序在对应安装范围内启动时，安装范围向该所关联的应用程序提供原生资源的唯一视图。在这些实施例中，该唯一视图专用于每一应用程序的操作需求。应用程序还可以在系统范围 256 中启动，也就是它们可以不和隔离范围有关联。这使得在隔离环境 260 中可以选择执行诸如 Internet Explorer 的操作系统 210 应用程序以及第三方应用程序。

[0081] 不考虑应用程序安装在何处而在隔离范围中启动应用程序的这种能力减轻了应用程序兼容性和应用程序社交性问题，而不需要在隔离范围中分开安装应用程序。在不同隔离范围中选择性地启动所安装的应用程序的这种能力使得需要帮助应用程序（诸如 Microsoft **Word®**，Notepad 等）的应用程序有能力使那些帮助应用程序使用同样的规则集启动。此外，在多个隔离环境中启动应用程序的能力允许在隔离的应用程序和普通应用程序之间更好的集成。

[0082] 图 2C 中所示为用于将应用程序和安装范围相关联的方法 2003。该方法 2003 包括在暂停状态启动进程（步骤 2004），并且获取和所期望的安装范围相关联的规则（步骤 2006）。用于应用程序和所获取的应用程序规则的标识符保存在存储器元件中（步骤 2008）并且重新启动暂停的应用程序（步骤 2010）。应用程序做出的用于访问原生资源的随后调用被拦截或者钩挂（步骤 2012），并且和进程标识符相关联的规则（如果存在的话）被用于虚拟化对所请求的资源的访问（步骤 2014）。

[0083] 仍旧参考图 2C，并且更详细地，在暂停状态启动应用程序（步骤 2004）。在一些实施例中，定制的启动程序被用于完成在暂停状态启动应用程序的任务。这些实施例可以包括特别设计来在选择的安装范围内启动应用程序的启动程序。方法 2003 的其他实施例包括将例如通过命令行选项接受所期望的安装范围的规范作为输入的启动程序。在方法 2003 的一个实施例中，在暂停状态启动的应用程序是软件应用程序，而在其他实施例中该应用程序可以是进程、软件命令、例程或者可执行程序。

[0084] 在方法 2003 的实施例中，获取和所期望的安装范围相关联的规则（步骤 2006）。一些实施例从诸如硬盘驱动器或者其他固态存储器元件的永久存储元件中获取该规则。其他实施例是从以下任一位置获取的规则：关联数据库，平面文件 (flat file database) 数据库，树形结构数据库，二叉树结构，自定义配置的数据结构，临时存储器位置，或者任一其它

类型的永久数据结构。其他实施例并不获取规则，而是从临时存储装置（未示）中获取用户指定的命令。

[0085] 在一个实施例中，应用程序标识符和所获取的规则保存在存储器元件中（步骤2008）。在一个实施例中，其中应用程序是进程，所获取的规则和进程 id(PID) 保存在存储器中。当提供了用于接收关于新进程建立的操作系统 210 消息的内核模式驱动时，方法2003 的另一个实施例可以将应用程序标识符和规则保存在内核模式驱动的上下文中。在其他实施例中，提供文件系统过滤器驱动或者迷你过滤器来拦截原生资源请求。在这些实施例中，PID 和所获取的规则可以保存在过滤器中。在其他实施例中，通过用户模式钩子执行所有的拦截并且不保存应用程序标识符。在此实施例中，在应用程序初始化期间，由用户模式的钩子设备加载该规则，并且由于规则关联完全在应用程序中执行，其它组件不需要了解应用到应用程序标识符上的规则。

[0086] 在方法 2003 的一个实施例中，重新启动暂停的应用程序（步骤 2010），并且应用程序做出的用于访问原生资源的任一随后调用都被拦截或者钩挂（步骤 2012）。包括规则的实施例使用和应用程序标识符相关联的规则来虚拟化对所请求资源的访问（步骤 2014）。在一些实施例中，文件系统过滤器驱动或者迷你过滤器拦截访问原生资源的请求并且确定和所拦截请求相关联的应用程序标识符是否已经和规则集相关联。在一个实施例中，当应用程序标识符和规则集相关联时，那些所关联的规则被用来对应用程序做出的用于访问原生资源的请求进行虚拟化。当没有规则和应用程序标识符相关联时，访问原生资源的请求直接通过而不做修改。在其他实施例中，动态链接库被载入新建立的应用程序中并且该库加载隔离规则。在另一些其他实施例中，内核模式技术（钩子，过滤器驱动，迷你过滤器）和用户模式技术二者被用来拦截访问原生资源的调用。对于文件系统过滤器驱动保存规则的实施例，该库可以加载来自文件系统过滤器驱动的规则。

[0087] 方法 2003 的实施例包括和安装范围相关联的应用程序的“子”应用程序，该安装范围还和它们的“父”应用程序的安装范围相关联。在这些实施例中，当建立子应用程序时通过内核模式驱动通知文件系统过滤器驱动来实现该方法 2003。在这些实施例中，文件系统过滤器驱动确定父进程的应用程序标识符和安装范围是否相关联。如果相关联，则文件系统过滤器驱动保存新近建立的子应用程序的应用程序标识符和父应用程序的安装范围之间的关联。在其他实施例中，文件系统过滤器驱动可以从系统直接调用，而不需要使用内核模式驱动。在其他实施例中，在和安装范围相关联的应用程序中，操作系统 210 用来钩挂或者拦截建立的新应用程序。当从这样的应用程序接收到建立新进程的请求时，保存新的子应用程序和父应用程序的安装范围之间的关联。

[0088] 提供原生资源的实例

[0089] 图 3A 中所示为将包括在范围中的资源视图聚集在虚拟范围 262 中的方法 401 的一个实施例。方法 401 包括拦截对原生资源列举的请求，该请求由第一应用所产生（步骤403）。列举原生资源的系统范围实例，并且在列举期间所遇到的所有资源均被加到虚拟化原生资源视图中（步骤 406）。可以列举在最后一个安装范围中的原生资源（步骤 409），并且调用“所列举的资源实例增加和替代”子例程（步骤 412）。做出关于附加的安装范围是否包括在隔离环境中的决定（步骤 415）。当确定在隔离环境中存在附加的安装范围时，列举下一个安装范围中的原生资源（步骤 418）并且调用“所列举的资源实例增加和替代”子例

程（步骤 421）。再次做出关于附加的安装范围是否包括在隔离环境中的决定（步骤 415）。如果在隔离环境中不存在附加的安装范围，则可以列举用户范围中的原生资源的实例（步骤 424）并且调用“所列举的资源实例增加和替代”子例程（步骤 427）。随后将虚拟化原生资源视图递送到发出请求的应用程序（步骤 430）。

[0090] 继续参考图 3A 并且更详细地，在方法 401 的一个实施例中，由驱动执行方法 401。在一些实施例中，该驱动可以是以下任一种：文件系统过滤器驱动，迷你过滤器、驱动或者执行此处描述方法的上述任一其它类型的虚拟对象。方法 401 的其他实施例可以包括应用程序、软件程序、其它软件命令集所执行的方法 401。贯穿全文可以理解在任何时候所提及的总体称作的“范围”在子范围存在的时候也指子范围。

[0091] 在方法 401 的一个实施例中，由驱动拦截在计算机器 200 上执行的应用程序所产生的请求（步骤 403）。在一个实施例中，该请求是对于原生资源的列举的请求。而在方法 401 的一个实施例中，该请求可以是对于单个原生资源的请求，在其他实施例中，该请求是对超过一个的原生资源的请求。

[0092] 方法 401 列举原生资源的系统范围实例，并且将所遇到的所有原生资源增加到包括在虚拟范围 262 中的虚拟化原生资源视图中（步骤 406）。在方法 401 的一个是实施例中，驱动列举了包括在系统范围 256 中的原生资源的实例。在此实施例中，所列举的资源增加到包括在虚拟范围 262 中的虚拟化原生资源视图。在其他实施例中，当系统根目录 216 包括超过一个的系统范围时，方法 401 调用“所列举的资源实例增加和替代”子例程，而不是增加所列举的原生资源的系统范围实例。在此实施例中，“列举的资源实例增加和替代”子例程在超过一个的系统范围上循环执行并且增加和替代虚拟化原生资源视图中的原生资源实例。在一些实施例中，当在系统范围中没有发现所请求的原生资源的实例时，方法 401 行进到列举原生资源的最后一个安装范围实例（步骤 409）。方法 401 的又一些其他实施例包括仅列举和所请求资源有关的那些原生资源，而其他实施例可以包括列举仅对应于所请求原生资源的原生资源，并且另一些其他实施例可以包括列举所有可用的原生资源，而不考虑哪个原生资源被请求。

[0093] 在方法 401 的一个实施例中，列举了最后一个安装范围中的原生资源实例（步骤 409）。在此实施例中，最后一个安装范围是处于最接近系统范围的安装范围。在另一种方式中，最后一个安装范围是基础应用实例安装在其上的安装范围。该基础应用实例是由安装在安装根目录 214 中的其他安装范围内的其他应用程序和应用程序实例所共享的应用程序。在方法 401 的该实施例中，随后的安装范围将引用该基础应用程序的应用程序安装到其上。方法 401 的其他实施例在第一安装范围中列举原生资源实例，而其它实施例列举包括在安装根目录 214 中的安装范围中的原生资源实例。原生资源实例的列举意味着从一个范围获取原生资源的所有实例。方法 401 的一些实施例可以从最后一个安装范围获取、获得或者得到原生资源。

[0094] 在方法 401 的一个实施例中，调用“所列举的资源实例增加和替代”子例程。图 3B 中示出“所列举的资源实例增加和替代”子例程 451 的实施例。该子例程 451 在多个范围上循环执行，以将还没有包括在虚拟化原生资源视图中的原生资源实例增加到该虚拟化视图中，并且使用新近发现的原生资源实例来替代虚拟化视图中的那些原生资源实例。方法 401 的其他实施例可以将子例程 451 包括在主方法 401 中，或者可以将该子例程 451 的一部

分包括在主方法 401 中。

[0095] 可以做出关于附加的安装范围是否包括在安装根目录 214 中的决定（步骤 415）。当确定出安装根目录 214 中存在附加的安装范围时，列举原生资源的用户范围实例（步骤 424）。方法 401 的其他实施例可以解决超过一个的安装范围的情况，从而方法 401 不再做出关于是否存在附加安装范围的决定，而是以对应于包括在安装根目录 214 中的安装范围的数量的预定次数列举安装范围原生资源实例。方法 401 的其他实施例包括列举在安装范围的其中一个中的原生资源实例并且不执行关于附加安装范围是否包括在安装根目录 214 中的进一步检查。

[0096] 在方法 401 的一个实施例中，当确定存在附加的安装范围时，列举下一个安装范围中的原生资源实例（步骤 418）。最后一个安装范围是指最后一个安装范围的下一个安装范围。例如，如果安装根目录 214 包括从一到三顺序排列的三个安装范围，第三个安装范围最接近系统范围，则第三个安装范围可以是最后一个安装范围。在此例中，第二安装范围可以是所查询的下一个安装范围。这意味着第二安装范围中的原生资源实例可以是待列举的下一个原生资源实例集。进一步参考此例，第一安装范围可以包括待列举的最后一个原生资源实例集。

[0097] 调用“所列举的资源实例增加和替代”子例程 451（步骤 421），并且下一个安装范围中所列举的资源被增加到虚拟化原生资源视图，或者插入到虚拟化原生资源视图中以代替已经包括在虚拟化原生资源视图中的原生资源实例。在方法 401 的一个实施例中，做出关于安装根目录 214 中是否包括附加安装范围的决定（步骤 415）。在方法 401 的此实施例中，做出关于附加安装范围是否包括在安装根目录 214 中的决定（步骤 415），列举包括在下一个安装范围中的原生资源的实例（步骤 418），并且调用“所列举的资源实例增加和替代”子例程 451（步骤 412），直到安装根目录 214 中包括的所有安装范围都循环处理过。当确定第一安装范围不具有所请求的原生资源的实例时，方法 401 的其他实施例可以仅在附加的安装范围上循环执行（未示）。方法 401 的又一些其他实施例根据用户输入或者系统需求可以仅在安装范围的一部分上循环执行。

[0098] 如果确定不存在附加的安装范围，则列举用户范围中的原生资源的实例（步骤 424）。在此实施例中，在单个用户范围中搜索原生资源实例。其他实施例包括类似用于安装范围的循环（步骤 415- 步骤 421），以在用户根目录 212 包括多个用户范围时在多个用户范围 212 上循环执行。在方法 401 的一个实施例中，用户范围中的原生资源实例的列举包括列举对应于所请求原生资源的单个原生资源实例，或者在其他实施例中，可以包括列举和所请求原生资源相关的多个原生资源实例。

[0099] 方法 401 的实施例调用“所列举的资源实例增加和替代”子例程 451（步骤 427）以增加在用户范围中标识的所列举的原生资源实例，或者使用在用户范围中标识的所列举的原生资源实例来替代已经存在的原生资源实例。其他实施例包括的方法 401 将子例程 451 中的步骤合并到主方法 401 中。其他实施例可以将在用户范围中找到的任一原生资源实例返回到请求的应用程序。在此实施例中，虚拟化的原生资源视图不显示给应用程序，而是用户范围内的原生资源实例可以显示给请求的应用程序。假定包括在用户范围中的原生资源的那些实例优先于从安装范围或者系统范围获取的那些原生资源，这样的实施例可以运行方法 401。

[0100] 在方法 401 的一个实施例中，将虚拟化的原生资源视图递送到发出请求的应用程序（步骤 430）。在一个实施例中，该虚拟化视图可以包括原生资源的所有发现的实例。其他实施例将在最高级范围中找到的原生资源的单个实例返回给请求的应用程序。在此实施例中，根据类似于用户范围、第一安装范围、随后的安装范围、最后一个安装范围和系统范围的优先顺序来确定范围等级，其中用户范围具有最高优先级并且系统范围具有最低优先级。其他实施例可以包括和上述优先顺序相反的优先顺序。其他实施例可以包括用户限定的或者系统限定的优先顺序，或者和上述限定顺序不相同的任一其它优先顺序。在一个实施例中，将在最低优先级的范围中找到的原生资源的实例返回到请求的应用程序。其它的实施例包括将包括在每一范围中的所有原生资源的整个聚集返回给请求的应用程序，其中聚集的原生资源实例对应于包括在预定范围中的原生资源实例。

[0101] 图 3B 中示出“所列举的资源实例增加和替代”子例程 451 的实施例。该方法 451 确定来自特定范围的所列举的资源实例是否包括没有包括在虚拟范围 262 中的虚拟化原生资源视图内的资源（步骤 440）。当确定所列举的资源实例尚未包括在虚拟化原生资源视图中时，所列举的资源实例随后被增加到虚拟化原生资源视图中（步骤 443）。当确定所列举的资源实例已经包括在虚拟化原生资源视图中（步骤 440），或者当所列举的原生资源实例增加到虚拟化原生资源视图（步骤 443）时，做出一个决定来验证所列举的资源实例包括已经在虚拟化原生资源视图中包括的资源实例（步骤 446）。如果确定所列举资源实例不包括已经在虚拟化原生资源视图中包括的资源实例，则退出“所列举的资源实例增加和替代”子例程（步骤 452）并且重新启动主方法 401。如果确定所列举资源实例确实包括已经在虚拟化原生资源视图中包括的资源实例，则使用所列举的原生资源实例来替代虚拟化原生资源视图中与在列举资源中识别出的原生资源相同的那些资源（步骤 449）。随后退出“所列举的资源实例增加和替代”子例程（步骤 452）并且重新启动主方法 401。

[0102] 在一个实施例中，可以执行概念上类似的方法来列举安装范围或者包括超过一个的子范围的其他范围。在此实施例中，列举单个子范围，其中在聚集的资源视图中，来自更高级子范围的资源替代来自更低级子范围的匹配实例。

[0103] 继续参考图 3B 并且更详细地，所列举的原生资源实例从主方法 401 传递到子例程 451（步骤 412、步骤 421、步骤 427）并且确定所列举的原生资源实例是否包括在虚拟化原生资源视图中（步骤 440）。换句话说，在此时实施例中，当在虚拟化原生资源视图中已经包括了由虚拟名称或者大体类似于虚拟名称的其他形式的标识或者对于列举的原生资源实例给定的标识符识别出的原生资源的实例时，可以确定所列举的原生资源实例已经存在于虚拟化原生资源视图中。其他实施例可以包括将与包括在虚拟化原生资源视图中的原生资源实例相关联的虚拟名称或者其他识别规则和与从主方法 401 返回到子例程 451 的列举的原生资源实例相关联的虚拟名称或者其他识别原则相比较。其他实施例可以包括在整个虚拟化原生资源视图上搜索以识别出大体类似于列举的原生资源实例的原生资源。

[0104] 在一个实施例中，当确定所列举的资源实例不包括在虚拟化原生资源视图中时，所列举的资源实例随后增加到虚拟化原生资源视图（步骤 443）。在此实施例中，将所列举的原生资源实例增加到虚拟化原生资源视图的操作可以包括为所列举原生资源实例命名一个虚拟名称，或者以其他方式经由标识符来识别所列举的原生资源实例，其中该标识符用于将所列举原生资源实例的类型和配置表示给驱动或者其他对象。子例程的其他实施例

将所列举的原生资源增加到虚拟化原生资源视图，而不考虑原生资源的实例是否已经包括在虚拟化资源视图中。在这样的实施例中，随后的应用程序可以滤除完全相同的原生资源实例，或者虚拟化资源视图可以被配置为仅显示相关的原生资源实例给请求的应用程序。其他实施例可以包括不将新的资源增加到虚拟化原生资源视图的子例程 451。

[0105] 当确定列举的资源实例已经存在于虚拟化原生资源视图中时，进一步确定所列举的原生资源实例是否包括已经在虚拟化原生资源视图中包括的资源实例（步骤 446）。在此实施例中，进行进一步的确定以验证资源的实例大体类型相似和 / 或具有大体相似的标识符或者虚拟名称。其他实施例可以不包括关于所列举的资源实例是否已经包括在虚拟化原生资源视图中的第二确定，而可以从所列举的资源实例是否已经包括在虚拟化原生资源视图中的确定（步骤 440）进行到使用新近列举的原生资源实例来替代虚拟化原生资源视图中的资源（步骤 449）。

[0106] 在一个实施例中，当确定所列举资源实例已经存在于虚拟化原生资源视图中时，用新近列举的原生资源实例来替代该资源（步骤 449）。在此实施例中，已经包括在虚拟化原生资源视图中的原生资源实例具有比随后增加的原生资源实例更低的优先级。换句话说，由于在增加来自较高优先级范围（如用户范围）的原生资源实例之前，增加来自较低优先级范围（如系统范围 256）的原生资源实例，所以随后增加的原生资源实例比已经存在的原生资源实例具有更高的优先级。这样的一个例子可以是类型 A 资源的原生资源 A 的实例，其中原生资源 A 的实例是从系统范围 256 获取并且增加到虚拟化原生资源视图。在此例中，在用户范围中识别出原生资源 A 的随后的实例。因为用户范围内的原生资源指示用户指定的设置，其与本质上是在所有用户会话上共同的通用资源的系统原生资源相比是更期望的，所以用户范围内的那些原生资源具有比系统范围中的那些原生资源更高的优先级。进一步参考此例，来自用户范围的原生资源 A 的第二实例增加到虚拟化原生资源视图，并且来自系统范围的第一实例从虚拟化原生资源视图中删除。返回来自用户范围而不是系统范围的原生资源可以确保由用户创建的以及与用户范围相关联的设置可以在下一次用户执行应用时为该用户所用。换句话说，在此例中，使用较高优先级的原生资源实例替代较低优先级原生资源实例保留了用户设置。另一个例子可以包括系统范围和安装范围，其中使用原生资源的安装范围实例替代原生资源的系统范围实例保留了应用程序设置和配置，使得下一次执行该应用程序时，该设置和配置被保留。在其他实施例中，新近列举的原生资源实例增加到虚拟原生资源视图，并且之后应用过滤器来移除较低优先级原生资源实例。其他实施例包括当列举随后的并且大体类似的原生资源实例时重新配置原生资源的实例。在此实施例中，可以重新配置现存的原生资源实例，以使其类似于新近列举的原生资源实例。其他实施例所包括的子例程 451 在识别出这样的共同资源时不使用所列举的资源列表中的那些资源替代虚拟化原生资源视图中的那些资源。其他实施例包括的子例程 451 仅替代那些标以许可的那些资源实例，该许可表示允许使用较高或者较低的优先级的随后的资源替代该资源。其他实施例包括当所列举资源不同于虚拟化原生资源时替代共同的资源的子例程 451。

[0107] 在一些实施例中，当子例程 451 确定不存在包括在虚拟化原生资源视图中且对应于所列举的原生资源实例的原生资源实例时，子例程退出（步骤 452）。子例程 451 的其他实施例可以包括退出子例程（步骤 452）并且返回主方法 401。子例程 451 的其他实施例退

出子例程（步骤 452）和主方法 401。在子例程 451 的一个实施例中，子例程退出并且不返回主方法 401。

[0108] 尽管图 3A 和图 3B 实现了用于聚集原生资源视图的方法 401 和子例程 451，但是其他实施例可以包括方法 401，其中子例程 451 包含在方法 401 中。另一些其他实施例可以包括进一步分为附加子例程的方法 401。

[0109] 图 4 示出在对超过一个的范围进行聚集期间建立并且显示给在计算机器 200 上执行的应用程序的一个实施例。虚拟范围 262 的建立和显示还减弱了由于应用程序非兼容性和应用程序非社交性引发的问题。包含在虚拟范围 262 中的是用户范围 282、第一安装范围 284A、第二安装范围 284B、最后一个安装范围 284N 和系统范围 286，。在每一个范围中是一组原生资源 258’、258”、258”’、258””、258”””。

[0110] 进一步参考图 4 并且更详细地，虚拟范围 262 聚集修改的原生资源视图 258 的各个单个范围。所修改的资源视图 258 的聚集还建立一个显示给应用程序的可用原生资源的聚集视图。在此实施例中所示的虚拟范围 262 将系统范围 286 和安装范围 284A-284N 与用户范围 282 组合在一起。其他实施例可以将附加的范围（未示）和用户范围 286、安装范围 284A-284N 以及系统范围 286 组合在一起。另一些其他实施例可以包括用户范围 282、安装范围 284A-284N 以及系统范围 286 的任意组合，所包括的组合中虚拟范围 262 包括单类型的范围或者一个或者多种类型的范围。其他实施例包括列举并且聚集从用户范围 282 向下到系统范围 286 的各个原生资源的实例的虚拟范围 262。在此实施例中，系统范围 286 中的那些原生资源实例可以具有比安装范围 284A-284N 中原生资源实例和用户范围 282 中原生资源实例更高的优先级。其他实施例可以放弃建立虚拟范围 262，而是响应于应用程序请求并且使用一系列查询和响应来执行对所请求的原生资源的大体同时的获取，而不是列举资源并将其汇编到包括在虚拟范围 262 中的虚拟化原生资源视图中。

[0111] 在一个实施例中，每个所包括的范围具有一组修改后的原生资源实例 258。当执行在计算机器 200 上的应用程序请求原生资源时，该应用程序被给予大体类似于图 4 中所示虚拟范围的虚拟范围。在此范围中，用户专用的原生资源 258’和应用程序专用的原生资源视图 258”-258”” 以及系统专用的原生资源视图 258””’ 相组合，以建立原生资源的单个虚拟化原生资源视图。在此实施例中，用户指定的资源视图 258’优先于应用程序专用的资源视图 258”-258”’，而应用程序专用的资源视图 258”-258”” 优先于系统专用的资源视图 258””’。为优先于低级范围中的原生资源视图 258，意味着如果存在对于高级范围的原生资源视图和低级范围的原生资源视图二者皆共有的单个设置，则和高级范围相关联的设置可以包括在虚拟范围 262 中。高级和低级范围的概念可以解释为一个存在多个安装范围 230 的实施例，其中该多个安装范围基于每个范围与之前和之后范围相关的状态按照从第一个到最后一个的顺序排级。例如，在此实施例中，如果存在三个范围，则最后的范围的级别低于第二范围，第二范围的级别低于第一范围，并且第一范围的级别比第二范围和第三范围都高。在包括高级和低级范围的虚拟范围 262 的实施例中，在低级范围中对原生资源的修改对于和该低级范围相关联的所有高级范围而言都是可用的，但是在高级范围中对资源的修改对于和该高级范围相关的所有低级范围而言是不可用的。这些概念可以延伸到的实施例可以使用子范围，使得对包括在用户子范围中的原生资源视图的修改对于代表与用户子范围相关联的用户或用户组执行的、与可应用的安装子范围相关联的所有应用程序而言都

是共同的。

[0112] 如果应用程序试图打开原生资源的现有实例而不意图修改该资源，则返回到应用程序的特定实例是在虚拟范围 262 中找到的实例，或者等价地是将在对所请求资源的父辈进行虚拟化列举中出现的实例。从隔离环境 260 的观点来看，该应用程序被认为是请求打开“虚拟资源”并且用来满足该请求的原生资源的特定实例被认为是对应于所请求资源的“文本资源 (literal resource)”。

[0113] 如果代表用户执行的应用程序试图打开资源并且指示这样做意于修改该资源，则由于安装范围和系统范围中的资源对于代表其它用户执行的各个应用程序而言是共有的，因而该应用程序实例通常被给予该资源的专用 (private) 备份用于修改。典型地，做出该资源的用户范围备份，除非用户范围实例已经存在。由虚拟范围提供的聚集视图的定义是指将安装范围或者系统范围资源复制到用户范围的行为不改变虚拟范围 262 为所考虑的用户和应用程序、任何其它用户、或者任何其它应用程序实例提供的聚集视图。由代表用户执行的应用程序实例对备份的资源进行的随后修改不会影响不共享同一用户范围的任一其它应用程序实例的聚集视图。换句话说，那些修改不会改变用于其他用户或者用于和同一安装范围不关联的应用程序实例的原生资源的聚集视图。

[0114] 图 5 中示出将原生资源的实例返回到发出请求的应用程序的方法。该实施例包括方法 351，其中在拦截到在用户会话中执行的应用程序所产生的对原生资源的请求之后，大体同时响应该请求（步骤 353）。当拦截到原生资源请求时（步骤 353），在第一安装范围内执行对于对应于所请求原生资源的原生资源的实例的搜索（步骤 356）。确定是否找到原生资源的对应实例（步骤 359），如果找到该资源的实例，则从第一安装范围取回所找到的原生资源的实例（步骤 362）。当没有找到原生资源时，则执行关于第一安装范围和第二安装范围之间的关联的搜索（步骤 365）。确定是否找到第一安装范围和第二安装范围之间的关联（步骤 368），并且如果没有找到关联，则在系统范围中执行对于对应于所请求原生资源的原生资源的实例的搜索（步骤 383）。如果找到关联，则在第二安装范围中执行对于对应于应用程序所请求的原生资源的原生资源实例的搜索（步骤 371）。确定在第二安装范围中是否找到对应于所请求原生资源的原生资源的实例（步骤 374），并且如果没有找到对应的资源，则在系统范围内执行对于对应所请求原生资源的原生资源的实例的搜索（步骤 386），如果在第二安装范围中找到对应原生资源的实例，则从第二安装范围中获取对应于所请求原生资源的原生资源的实例（步骤 377）。当从第一安装范围或者所关联的第二安装范围获取原生资源的对应的实例时，将所获取的资源实例包括在所发送的针对应用程序做出的对于原生资源的请求的响应中（步骤 380）。

[0115] 继续参考图 5 并且更详细地，在所示实施例中，方法 351 包括拦截到在用户会话中执行的应用程序所产生的请求（步骤 353）。在此实施例中，由单个应用程序做出对原生资源的请求。其他实施例包括由一个或者多个应用程序做出对一个或者多个原生资源的请求。另外的实施例包括具有被配置为拦截应用程序请求和执行方法 351 的过滤器驱动、驱动或者其它虚拟对象的操作系统 210。

[0116] 在一个实施例中，在第一安装范围内执行对于与应用程序所请求原生资源相对应的原生资源实例的搜索（步骤 356）。第一安装范围是相对于包括在隔离环境 260 中的所有其他安装范围而言具有最高优先级的安装范围。例如，如果在隔离环境 260 中包括三个安

装范围，则第一安装范围可以是相对于其他两个安装范围具有最高优先级的范围。在此例中，最后一个范围可以是具有与其他两个安装范围的每一个相比更低优先级的安装范围。在此例中，优先级是决定是否将一个范围中的原生资源，而不是将包括在其他范围中的资源显示给发出请求的应用程序的决定因素。在此例中，包括在第一安装范围中的资源可以采用高于在第二安装范围或者最后一个安装范围中资源的优先级，这意味着为请求的应用程序示出的是包括在第一安装范围中的原生资源的实例，而不是包括在第二安装范围或者最后一个安装范围中的原生资源的实例。

[0117] 在一个实施例中，确定在第一安装范围中是否找到对应的原生资源（步骤 359）。如果找到对应的原生资源，则从第一安装范围取回该所找到的原生资源的实例（步骤 362）。在一些实施例中，该原生资源的实例可以被大体上立即返回给请求的应用程序，或者可以进行另外的搜索来找到所请求资源的额外实例。

[0118] 在一个实施例中，如果在第一安装范围中没有找到原生资源的对应实例，则执行对于第一安装范围和第二安装范围之间的关联的搜索（步骤 365）。实施例包括对于每个安装范围所隔离的各个应用程序搜索用于指示第一安装范围关联于第二安装范围的关联。在一个实施例中，在第一安装范围中隔离的第一应用程序可以和在第二安装范围中隔离的第二应用程序相互作用。与将每个应用程序合并到单个安装范围中不同，每个应用程序在各自的安装范围中执行并且在安装范围间建立的每个关联下各应用程序互相作用。在一些实施例中，安装范围之间的关联可以导致安装范围在隔离环境 260 中一起相互作用，而其他实施例可以导致安装范围在隔离环境 260 中基于各种具体的情况相互作用。

[0119] 确定是否找到第一安装范围和第二安装范围之间的关联（步骤 368），并且在一个实施例中，该方法随后在第二安装范围中搜索对应的原生资源（步骤 371）。虽然图 5 中描述的实施例说明了对于第一安装范围和第二安装范围之间的关联的搜索，但是其他事实可以包括对于一个或者多个安装范围之间的附加关联的搜索，例如：第二安装范围和第三安装范围之间的关联，第三安装范围和第四安装范围之间的关联，以及第“n”安装范围和第“n+1”安装范围之间的关联，其中 n 是大于 2 的任一数字。如果在第一安装范围和附加的安装范围之间没有找到关联，则该方法识别系统范围中的候选资源并且在系统范围中搜索以验证候选资源的存在（步骤 383）。

[0120] 在一个实施例中，确定在第二安装范围中是否存在对应的原生资源的实例（步骤 374）。在一个实施例中，如果没有找到资源实例，则方法 351 在系统范围内搜索所请求原生资源的对应实例（步骤 386）。在一个实施例中，从第二安装范围取回所找到的原生资源的实例（步骤 377），并且在将应用程序请求的响应发送给应用程序时将该原生资源实例发送给请求的应用程序（步骤 380）。

[0121] 另一个实施例包括配置为将任一应用程序实例和任一安装范围相关联的计算机器 200，其不考虑应用程序是安装到该应用程序隔离范围，还是安装到另一个应用程序隔离范围或者不安装到应用程序隔离范围。未安装到特定应用程序范围的应用程序可以代表用户在安装范围和相应的用户范围的上下文中执行，这是因为这些应用程序的原生资源可经由用户范围、安装范围和系统范围形成的聚集的虚拟范围 262 为这些应用程序所用。在期望以隔离方式运行应用程序的时候，该计算机器 200 为直接安装在系统范围中的应用程序提供在隔离环境 260 中运行的能力，而不需要将该应用程序单独安装在隔离环境 260 中。这

也为直接安装在系统范围 256 中的应用程序提供了可用作任一范围的上下文中的帮助应用程序的能力。

[0122] 包括组成执行的应用程序的所有进程的每一个应用程序实例可以与零个或者一个安装范围相关联，以及通过恰当扩展与零个或者一个对应的用户范围相关联。当确定哪个规则（如果存在的话）应用到资源请求时，由规则引擎使用该关联。该关联不必到应用程序所安装的安装范围（如果有的话）。安装在一个范围中的许多应用程序在不同范围或者不在范围中运行时由于它们不能找到所需的原生资源，而不能正确发挥功用。然而，由于虚拟范围 262 是包括系统范围的资源视图的聚集，安装在系统范围中的应用程序通常可以在任意安装范围中正确运行。这意味着帮助程序以及进程外 COM 服务可以通过代表用户在特定范围中执行的应用程序来调用和执行。

[0123] 在一些实施例中，安装在系统范围中的应用程序在一范围内执行，目的是识别出对计算机文件和配置设置进行了何种改变作为该执行的结果。当所有受影响的文件和配置设置都隔离在用户范围中时，这些文件和配置设置是很容易识别的。在一些这样的实施例中，这样使用是为了报告该应用程序对文件和配置设置做出的改变。在一些实施例中，在应用程序执行的最后，删除文件和配置设置，这样可以有效确保对计算机文件和配置设置进行的改变没有保存作为应用程序执行的结果。在其他实施例中，在应用程序执行的最后，选择性地删除或者不删除文件和配置设置，这有效确保仅有对计算机文件和配置设置作出的某些改变保存作为应用程序执行的结果。

[0124] 虚拟化机制综述

[0125] 图 6A 中所示是在执行模式中虚拟化访问原生资源的实施例。在以下详细描述中，访问模式和安装模式是可区分的。总的来说，拦截或者接收访问原生资源的请求（步骤 502）。该请求识别出访问所寻找的原生资源。确定关于如何处理所接收的访问请求的可应用的规则（步骤 504），并且进一步做出关于应该采取何种动作的决定（步骤 505）。如果该规则指示该请求应该被忽略，则将文本资源名称设置成等于虚拟资源名称（步骤 507），传递访问请求而不修改系统根目录（步骤 506），并且将结果返回请求者（步骤 510）。如果该规则指示访问请求应该被重定向或者隔离，则识别满足该请求的资源的文本实例（步骤 508），将对于文本资源的修改或者替换请求传递到系统层（步骤 506）并且将该结果返回到请求者（步骤 510）。

[0126] 继续参考图 6A 并且更详细地，拦截或者接收识别原生资源的请求（步骤 502）。在一些实施例中，对于发出对原生资源的请求的应用程序，由操作系统 210 提供的“钩子”函数拦截该对原生资源的请求。在一个实施例中，经由动态链接库 (DLL) 实现使用“钩子”函数的拦截，该 DLL 载入到由操作系统 210 建立的新进程的地址空间中，并且在其初始化例程期间执行钩挂功能。在一些实施例中，使用操作系统 210 提供的工具将 DLL 载入每个进程，或者在其他实施例中通过将 DLL 列表修改为可执行映像中包括的导入 (import)，其中 DLL 列表或者在磁盘文件中，或者在存储器中作为进程从磁盘加载的可执行映像。其他实施例使用服务程序、驱动或者后台程序执行“钩子”函数。在其他实施例中，操作系统 210 提供的包括共享库和可执行文件的可执行映像可以被修改或者打补丁，用来提供功能钩子或者直接实现本发明的逻辑。对于操作系统是微软 WINDIWS 家族操作系统的成员的特定实施例，通过钩挂系统服务分派表的内核模式驱动来执行拦截。在其他实施例中，操作系统 210 可

以提供允许第三方来钩挂访问原生资源的请求的工具。在一些这样的实施例中，操作系统 210 可以经由应用程序接口 (API) 或者调试工具来提供这种工具。

[0127] 在其他实施例中，通过和原生资源相关联的驱动堆栈或者处理程序堆栈中的过滤器来拦截原生资源请求。例如，微软 WINDIWS 家族操作系统的一些成员提供将第三方过滤器驱动或者迷你过滤器插入到文件系统驱动堆栈中的性能，并且文件系统过滤器驱动或者迷你过滤器可以被用来提供下述的隔离功能。本发明的其他实施例包括直接包括本发明的逻辑的文件系统实现方案。此外，操作系统 210 可以被重写以直接提供下述功能。在一些实施例中，可以同时使用上面列出的用来拦截或者接收对于资源的请求的一些或者全部方法的组合。

[0128] 在许多实施例中，仅钩挂或者拦截用来打开现有原生资源或者建立新的原生资源的请求。在这些实施例中，对原生资源的初始访问是导致资源被虚拟化的访问。在初始访问之后，请求的应用程序可以使用操作系统 210 提供的、用于直接识别文本资源的句柄、或者指针或者其他标识符就该虚拟化资源与操作系统 210 通信。在其他实施例中，操作虚拟化原生资源的其它类型的请求也被钩挂或者拦截。在一些这样的实施例中，应用打开或者建立虚拟资源的请求返回一个不直接识别文本资源的虚拟句柄，并且隔离环境 260 用于将对于虚拟句柄的随后请求翻译为对应的文本资源。在一些这样的实施例中，附加的虚拟化操作可以被延缓，直到证明是必要的。例如，将资源的专用 (private) 可修改备份提供给安装范围的操作可以被延缓，直到做出改变资源的请求时候，而不是在允许随后修改的模式中打开该资源的时候。

[0129] 在一些实施例中，一旦拦截或者接收到原生资源请求，确定关于如何处理特定请求的可应用的规则 (步骤 504)。在一些实施例中，该可应用的规则通过参考规则引擎、规则数据库或者包含使用诸如列表或者树结构的合适数据结构组织的规则的平面文件来确定。在其他实施例中，规则被给予一个优先级，其用于确定当应用两个或者多个规则时哪个规则被认为是可应用的规则。在许多实施例中，规则优先级包括在规则自身中，或者在其他实施例中，规则优先级可以嵌入到用来保存规则的数据结构中，例如规则优先级可以通过树结构中的规则的位置来指示。在这些实施例中，所确定的规则可以包括关于如何处理虚拟化资源请求的附加信息，例如将请求重定向到哪个文本资源。在特定实施例中，规则是包括过滤器字段、动作字段和数据字段的三元组。在此实施例中，过滤器字段包括用来匹配所接收的原生资源请求来确定该规则对于所请求的资源名称是否有效的过滤器。

[0130] 在一个实施例中，做出关于应该执行何种动作的决定 (步骤 505)。方法 503 的一些实施例在获取的规则信息中包含该动作信息。示例性的动作包括 :“忽略”,“重定向”,或者“隔离”。其他实施例包括数据字段，该数据字段包括关于当规则有效时将要采取的动作的附加信息，包括在规则有效时将要使用的功能。方法 503 的另一些其他实施例包括做出关于应该执行何种动作的决定 (步骤 505)，其中动作指令由用户输入产生。

[0131] 在许多实施例中，规则动作“忽略”意味着请求直接操作系统范围中所请求的原生资源。也就是，文本资源名称设置等于虚拟资源名称 (步骤 507)，并且该请求毫无改变地传递到系统根目录 (步骤 506)，并且该请求被实现成如同不存在隔离环境 260 那样。在此情况中，隔离环境 260 被认为具有“洞”，或者该请求被称为“穿透”请求。

[0132] 在一些实施例中，如果规则动作指示原生资源请求应该被重定向或者隔离，则识

别满足该请求的文本资源（步骤 508）。在许多实施例中，规则动作“重定向”是指原生资源请求直接操作系统范围原生资源，即使其与在请求中指定的资源不同。通过将在所确定的规则的数据字段中指定或者暗示的映射函数施加到所请求原生资源的名称上，可识别出该文本资源。在大部分一般情况中，文本原生资源可以位于系统范围 256 中的任一位置。举一个简单例子，规则 {prefix_match(" c:\temp\"，resource name)，REDIRECT，replace_prefix(" c:\temp\"，" d:\wutemp\"，resource name)} 可以将所请求的对文件 c:\temp\examples\dl.txt 的访问重定向到文本文件 (literal file)d:\wutemp\examples\dl.txt。包括在规则的数据字段中的映射函数和匹配函数还可以通过使用规则的表达式而被推广为用来支持更复杂的行为。一些实施例可以提供指定映射函数的能力，该映射函数将文本资源定位在用户隔离范围或者对于代表用户执行的应用程序可应用的子范围，或者安装范围或者对于该应用程序可应用的子范围中。其他的实施例可以提供指定映射函数的能力，该映射函数将文本资源定位在对于不同应用程序可应用的安装范围中，以提供隔离的应用程序之间可控形式的相互作用。在一些特定实施例中，“重定向”动作可以被配置来提供等价于“忽略”规则动作的行为。在这些实施例中，文本资源正是所请求的原生资源。当设定该条件时，隔离环境 260 可以称为具有“洞”，或者该请求可以称为“穿透”请求。

[0133] 规则动作“隔离”表示该请求操作使用正确的用户范围和安装范围识别的文本资源。也就是说，使用用户范围、安装范围、这两个范围或者一个也不使用来修改所请求的原生资源的标识符，从而确定文本资源的标识符。所识别的特定文本资源依赖于所请求的访问类型和所请求的原生资源的实例是已经存在于可应用的用户范围、可应用的安装范围还是系统范围 256。

[0134] 图 6B 描述过程 508 的实施例，包括当接收到打开原生资源并指示打开该资源的目的是进行修改的请求时为识别文本资源所采取的步骤（图 6A 中步骤 506）。简要地，确定是否存在所请求原生资源的用户范围实例，也就是在可应用的用户范围或者用户子范围中存在的实例（步骤 554）。如果存在，将用户范围实例识别为该请求的文本资源（步骤 506），并且该实例被打开并返回到请求者。如果用户范围实例不存在，则确定是否存在所请求原生资源的第一安装范围的实例（步骤 556）。如果第一安装范围的实例存在，则其被识别为“候选”资源实例（步骤 569），并且检查和候选实例相关联的许可数据来确定是否允许修改该实例（步骤 562）。如果不存在第一安装范围实例，则确定是否存在第一安装范围和第二安装范围之间的关联（步骤 557）。如果在第一安装范围和第二安装范围之间存在这样的关联，则确定是否存在所请求原生资源的第二安装范围的实例（步骤 559）。如果存在第二安装范围实例，将其识别为“候选”资源实例（步骤 569）。如果不存在第一安装范围和第二安装范围之间的关联，或者如果存在关联而第二安装范围的实例不存在，则确定是否存在所请求原生资源的系统范围的实例（步骤 558）。如果不存在系统范围的实例，则将错误状况返回到请求者，指示所请求的虚拟资源在虚拟范围中不存在（步骤 560）。然而，如果存在系统范围的实例，该实例被识别为候选资源实例（步骤 569），并且检查和候选实例相关联的许可数据以确定是否允许修改该实例（步骤 562）。如果否，则将错误状况返回给请求者（步骤 564），指示不允许修改虚拟资源。如果许可数据指示可以修改候选资源，则做出原生资源的候选实例的用户范围备份（步骤 570），用户范围的实例被识别为对于该请求的文本实例（步骤 506），并且被打开和返回给请求者。

[0135] 继续参考图 6B 并且更详细地，在过程 508 的一个实施例中，确定是否存在用户范围的资源，或者换句话说，在可应用的用户范围或者用户子范围中是否存在所请求的资源（步骤 554）。一些实施例中，可应用的用户范围或者子范围是和用于执行发出该请求的应用程序的用户会话相关联的范围。在所请求的资源是来自文件系统的文件的实施例中，该用户范围或者子范围可以是用户范围中存在的所有文件保存在其下的目录。在一些这样的实施例中，用户范围目录下的目录树结构反映所请求资源的路径。例如，如果所请求的文件是 c:\temp\test.txt 并且用户范围目录是 d:\user1\app1\，则到用户范围的文本文件的路径可以是 d:\user1\app1\c\temp\test.txt。在其他实施例中，到用户范围的文本文件的路径可以以原生命名约定来限定。例如，到用户范围的文本文件的路径可以是 d:\user1\app1\device\harddisk1\temp\test.txt。在其他实施例中，用户范围文件可以全部保存在具有选为唯一名称的单个目录中并且可以使用数据库来保存所请求文件名和保存在目录中的对应文本文件的名称之间的映射。在另一些其他实施例中，文本文件的内容可以保存在数据库中。在另一些其他实施例中，原生文件系统提供用于单个文件的工具以包括多个独立命名的“流”，并且用户范围的文件的内容保存为系统范围 256 中的相关文件的附加流。或者，文本文件可以保存在可以设计来优化磁盘使用或者其他感兴趣的标准的自定义文件系统中。

[0136] 如果用户范围资源实例不存在，则确定是否存在第一安装范围的资源，或者换句话说在第一安装范围内是否存在所请求的资源（步骤 556）。在一个实施例中，上述方法被用来作出该决定。例如，如果所请求的文件是 c:\temp\test.txt 并且第一安装范围目录是 e:\app1\，则到第一安装范围的文件的路径可以是 e:\app1\c\temp\test.txt。如上，其他实施例可以允许到第一安装范围的文件的路径以原生命名约定来保存。上述实施例还可以应用程序到第一安装范围。

[0137] 如果不存在第一安装范围的资源，则确定是否存在第一安装范围和第二安装范围之间的关联（步骤 557）。在许多实施例中，多个安装范围可以存在于隔离环境 260 中，意味着第一安装范围可以和一个或者多个附加的安装范围相关联，使得这些附加的安装范围的映像包括在和第一安装范围相关联的隔离环境 260 中。尽管图 6B 示出过程 508 仅检查和第二安装范围的关联，过程 508 的其他实施例可以包括检查一个或者多个附加的安装范围。在此实施例中，可以分析每个附加的安装范围以确定安装范围是否包括所请求资源的实例（类似于步骤 559）并且当决定结果为真时，所找到的资源可以被识别为候选资源（步骤 569）

[0138] 如果在第一安装范围和第二安装范围之间存在这样的关联，则确定第二安装范围是否包括所请求资源的实例（步骤 559）。在一个实施例中，可以使用上述方法做出该确定。例如，如果所请求的文件是 c:\temp\test.txt 并且第二安装范围目录是 e:\app1\，则到第二安装范围的文件的路径可以是 e:\app1\c\temp\test.txt。如上，其他实施例可以允许到第二安装范围的文件的路径以原生命名约定来保存。上述实施例还可以应用到第二安装范围。

[0139] 如果第二安装范围的资源不存在，或者不存在第一安装范围和第二安装范围之间的关联，则确定是否存在系统范围的资源，或者换句话说在系统范围中是否存在所请求的资源（步骤 558）。例如，如果所请求的文件是 c:\temp\test.txt，则到系统范围的文件的

路径是 c:\temp\test.txt。如果在系统范围中不存在所请求的资源，则将虚拟范围中不存在所请求资源的指示返回给请求者（步骤 560）。

[0140] 无论所请求资源的候选资源实例是位于多个安装范围之一中，还是位于系统范围中，确定是否允许修改候选资源实例（步骤 562）。例如，候选原生资源可能已经关联于用来指示该用户会话不允许修改该候选实例的原生许可数据。此外，规则引擎可以包括用来指示隔离环境 260 遵循或者无视针对资源的虚拟备份的原生许可数据的配置设置。在一些实施例中，对于一些虚拟资源，该规则可以指定修改将发生的范围，例如系统范围、安装范围或者子范围，或者用户范围或者子范围。在一些这样的实施例中，规则引擎可以根据所访问的资源的层次或类型来指定施加到虚拟化原生资源的子集的配置设置。在一些这样的实施例中，配置设置可以专用于每个虚拟化原生资源。在另一个例子中，规则引擎可以包括禁止或者允许修改特定文件类的配置数据，诸如可执行代码，MIME 类型或者如操作系统 210 限定的文件类型。

[0141] 如果确定该许可不允许修改候选资源实例，则将错误状况返回给请求者，指示不允许对虚拟资源进行写访问（步骤 564）。如果确定该许可允许修改候选资源实例，则将候选实例复制到合适的用户范围或者子范围（步骤 570）。对于在不同范围中保持所请求原生资源的逻辑分层结构的实施例，将资源的候选实例复制到用户范围（步骤 570）可能需要在用户范围中建立分层占位符。分层占位符是置于分层中以正确定位范围中复制的资源的节点。不保存数据的分层占位符被识别为占位符节点，并且当其不是返回请求者的文本资源时其“并不存在”。在一些实施例中，将节点标示为占位符节点是通过将这一事实记录在接附到该节点或者该节点的父节点，或者接附到系统层中的一些其它相关实体的元数据中来实现的。在其他实施例中，维持一个单独的占位符节点名称的知识库。

[0142] 在一些实施例中，该规则可以指定在特定范围处可以修改特定资源，诸如安装范围。在这些情况中，复制操作被延伸（步骤 570）以确定在所找到的范围和子范围处是否允许修改候选资源实例。如果否，则将候选资源实例复制到允许修改的范围或者子范围，其可以不是用户范围，并且新的备份被识别为文本资源实例（步骤 506）。如果是，则将候选资源实例识别为文本实例（步骤 506），并且打开该候选资源以及将结果返回给请求者（步骤 510）。

[0143] 返回参考图 6A，无论文本资源实例是位于用户范围（步骤 554）还是在复制到用户范围时被建立（步骤 570），其被打开（步骤 506）并且返回到请求者（步骤 510）。在一些实施例中，通过发布“打开”命令到操作系统 210 并且将来自操作系统 210 的对“打开”命令的响应返回到发出请求的应用程序，来实现打开资源实例并且将实例返回到发出请求的应用程序。

[0144] 如果代表用户执行的应用程序删除了原生资源，则提供给该应用程序作为虚拟范围 262 的原生资源的聚集视图反映了该删除。在一个实施例中，删除资源的请求是对于特殊类型修改的请求，即使该请求是通过完全消除该资源的存在来对其进行修改。概念上，删除资源的请求以在图 6A 中所概括的相似方式来进行，包括在图 6B 中所概述的文本资源的确定。然而，在支持删除操作的实施例中，识别文本资源实例（步骤 506）对于隔离资源和对于重定向或者忽略资源进行不同的操作。当动作是“重定向”或者“忽略”时，从系统范围删除文本资源。当动作是“隔离”时，“虚拟”删除该文本资源，或者换句话说，在用户范围内

记录其已经被删除的事实。所删除的节点不含有数据，被识别为删除，并且其和所有的子节点“不再存在”。也就是，如果其是以其他方式满足资源请求的资源或者资源的祖先，则“资源未发现”的错误被返回到请求者。以下概述部分是关于该过程的更具体的细节。在一些实施例中，将节点识别为删除的节点是通过在接附到节点、或者节点的父节点、或者系统根目录 216 中的一些其他相关实体的元数据中记录此事实来实现的。在其他实施例中，例如，在单独的子范围中维持所删除的节点名称的单独的知识库。

[0145] 将应用程序安装到安装范围

[0146] 在许多实施例中，如上描述的安装范围是其中一个应用程序的各相关实例独立于其他用户会话或者同等地代表所有可能的用户会话来共享资源的范围，其包括那些应用程序实例所建立的资源。实施例可以包括在应用程序安装到操作系统 210 上时所建立的这些资源的主类。一些实施例包括计算机 200，其中两个不兼容的应用程序不被允许安装在其上。在此实施例中，两个不同应用程序之间不兼容的问题可通过将至少一个应用程序安装到隔离环境 260 中来解决。为了实现将应用程序安装到隔离环境 260 中，安装方法的一个实施例允许安装范围或者和安装范围相关联的应用程序实例在“安装模式”中操作，使得安装范围或者应用程序实例可以支持应用程序的安装。在此实施例中，“安装模式”利用针对所选应用程序的安装程序，并且在假设安装程序应代表所有用户会话执行的情况下将此安装程序和安装范围相关联。在此实施例中，该安装范围相当于应用程序实例，如同安装范围是对于“所有用户”的用户范围，尽管对于该应用程序实例不存在有效的用户范围。

[0147] 图 6C 中示出在安装模式中操作的方法 508 的实施例，并且方法 508 被配置成在接收到用于打开原生资源的请求且该请求指示该资源被打开的目的是对其进行修改时来识别文本资源。首先确定在第一安装范围中是否存在资源实例（步骤 524）。如果在第一安装范围中不存在资源实例，则确定在第一安装范围和第二安装范围之间是否存在关联（步骤 526）。如果在第一安装范围和第二安装范围之间不存在关联，则确定在系统范围中是否存在资源实例（步骤 530）。如果在第一安装范围和第二安装范围之间存在关联，则确定在第二安装范围中是否存在资源实例（步骤 528）。如果确定在第二安装范围中存在资源实例，则文本实例被识别为在第二安装范围中找到的实例（步骤 531）。如果确定在第二安装范围中不存在资源实例，则确定在系统范围中是否存在所请求资源的实例（步骤 530）。如果在系统范围中不存在资源实例，则将“未找到”错误返回给请求的应用程序（步骤 532）。然而，如果在系统范围中找到所请求资源的实例，则将所请求资源的系统范围实例识别为候选资源（步骤 534）。随后确定和所识别的资源相关联的许可是否允许修改所识别的资源实例（步骤 536）。如果该许可不允许修改所识别的资源实例，则将“许可被拒绝”的错误返回给请求的应用程序（步骤 538）。然而，当许可允许修改所识别的资源实例时，则将候选资源复制到第一安装范围（步骤 540），并且文本资源是第一安装范围中的资源（步骤 542）。当确定第一安装范围中存在资源实例时（步骤 524），则文本资源是第一安装范围中的资源实例（步骤 542）。

[0148] 继续参考图 6C 并且更详细地，在方法 508 的一个实施例中，没有有效的用户范围，所以首先确定是否存在所请求的原生资源的第一安装范围实例（步骤 524）。如果第一安装范围实例存在，则将其识别为文本资源实例（步骤 542）。如果不存在第一安装范围实例，则确定在第一安装范围和第二安装范围之间是否存在关联（步骤 526）。如果确定存在关联，

则进一步确定是否存在第二安装范围实例（步骤528）。在方法508的其他实施例中，在超过两个的安装范围中搜索该资源的实例，并且确定在超过两个的安装范围之间是否存在关联。

[0149] 在方法508的实施例中，如果未找到在第一安装范围和第二安装范围之间的关联，则确定在系统范围中是否存在资源实例（步骤530）。如果确定在第二安装范围中存在资源实例，则将找到的资源被识别为文本资源（步骤531）。方法508的其他实施例可以将所识别的资源实例复制到第一安装范围（步骤540），并且将复制的实例识别为文本资源。

[0150] 在方法508的实施例中，当确定系统范围中识别为候选资源的资源（步骤534）已经被许可修改（步骤536）时，则将该资源复制到第一安装范围（步骤540）并且将其识别为文本资源（步骤542）。在方法508的一些实施例中，候选文件被复制到规则引擎限定的位置。例如，规则可以指定文件复制到应用程序隔离范围。在其他实施例中，规则可以指定文件应该复制到的特定的应用程序隔离子范围或者用户隔离子范围。没有出现在所请求的文件所复制到的隔离范围中的该文件的任一祖先被创建为该隔离范围中的占位符，以便正确定位分层中的复制的实例。

[0151] 图6D中所示为在安装模式中操作的方法596，使得方法596被配置为当接收到建立原生资源的请求时识别文本资源。简要地，当没有有效的用户范围时，首先确定是否存在所请求原生资源的第一安装范围实例（步骤548）。如果存在第一安装范围实例，则将指示该资源由于其已存在而不能建立的错误状况返回给请求者（步骤590）。如果不存在第一安装范围实例，则确定在第一安装范围和第二安装范围之间是否存在关联（步骤586）。如果在第一安装范围和第二安装范围之间存在关联，则确定是否存在所请求原生资源的系统范围实例（步骤588）。如果在第二安装范围中存在资源实例，则将指示该资源由于其已存在而不能建立的错误状况返回给请求者（步骤590）。如果在第二安装范围中不存在该资源的实例，则确定在系统范围中是否存在该资源的实例（步骤592）。当在系统范围内存在该资源的实例时，则将指示该资源由于其已存在而不能建立的错误状况返回给请求者（步骤590）。在一些实施例中，用来打开资源的请求可以指定任一现有的该资源的系统范围实例可以被重写。如果系统范围资源实例不存在，则这两个安装范围资源实例中的任一可以被识别为其被建立来满足该请求的文本实例（步骤594）。

[0152] 通过将图6B和图6C、图6D相比较，可以看到安装模式的操作和执行模式类似，仅是安装范围取代了用户范围。换句话说，对永久资源的修改（包括建立新的资源）发生在合适的安装范围，而不是合适的用户范围。此外，对现存隔离资源的虚拟化访问也忽略合适的用户范围并且在安装范围中开始搜索候选文本资源。

[0153] 安装范围以此方式操作来包含对现有资源的修改和建立新的资源存在两者其它情况。第一种，存在被配置为在无需用户层的情况下操作的隔离环境260，或者存在被配置为在无需用户范围的情况下操作的虚拟范围262。在此情况中，安装范围仅是可以隔离修改资源和新近建立资源的范围。第二种，用于管理特定虚拟资源集的规则可以指定它们被隔离到合适的安装范围中而不是合适的用户范围中。此外，这意味着经受该规则的资源的修改和建立被隔离在这些资源的修改和建立对于共享该范围的所有应用程序实例皆可见的合适的安装范围中，而不是隔离在这些资源的修改和建立仅对于执行那些应用程序实例的用户可见的用户范围中。

[0154] 在另一些其他实施例中，隔离环境 260 可以被配置为允许特定资源在系统范围中共享，也就是隔离环境 260 可以代表一个或者多个系统资源，如同不存在用户范围和安装范围那样。由于系统范围中共享的系统资源为所有应用程序和所有用户所共享，也即它们是全局对象，所以当它们以修改目的被访问时不被复制。

[0155] 图 7 中所示为用于将应用程序安装到隔离环境 260 的方法 301 的实施例。限定隔离环境 260（步骤 303），并且在第一应用程序和第一安装范围之间建立第一关联（步骤 306）。在建立第一关联后，确定该第一应用程序的执行是否需要第二应用程序的实例（步骤 309）。如果确定指示不需要安装附加的应用程序实例来在隔离环境 260 中安装该应用程序，则保存第一关联（步骤 330），并且使第一应用程序在隔离环境 260 中启动（步骤 327）。如果确定指示需要附加的应用程序实例来将应用程序安装到隔离环境中，则将第二应用程序的映像装到第二安装范围上（步骤 312）。建立第二应用程序和第二安装范围之间的第二关联（步骤 315）并且建立第一安装范围和第二安装范围之间的关联（步骤 318）。在第一安装范围与第二安装范围的聚集与第三安装范围之间建立第三关联（步骤 321）。随后在存储器元件中保存第一关联、第二关联和第三关联（步骤 324）。

[0156] 继续更详细地参考图 7，方法 301 的实施例包括在操作系统 210 中限定隔离环境 260（步骤 303）。方法 301 的其他实施例可以将隔离环境 260 限定在一个包括在计算机器 200 中但位于操作系统 210 之外的单个的存储器元件或者应用程序中。在方法 301 的实施例中，隔离环境 260 位于和计算机器 200 相通信但安装在计算机器 200 所安装位置的远程位置的第三计算机器（未示）上。在一些实施例中，隔离环境 260 没有被限定在操作系统 210 中，相反安装根目录 214 和用户根目录 212 被作为单独实体包括在计算机器 200 上。

[0157] 在方法 301 的一个实施例中，在第一应用程序和第一安装范围之间建立第一关联（步骤 306）。在此实施例中，使用对于第一应用程序和第一安装范围共有的标识符来建立第一关联。方法 301 的另一个实施例可以通过将条目输入到用于跟踪在应用程序和安装范围之间建立的关联的表、列表或者其他文件中来建立该关联。方法 301 的另一些其他实施例包括在计算机器 200 上保持一个虚拟对象，其代表第一应用程序和第一安装范围之间建立的关联。

[0158] 在方法 301 的一个实施例中，确定第一应用程序是否需要第二应用程序的存在以便正确执行（步骤 309）。在方法 301 的一些实施例中，应用程序可以要求另一个应用程序的存在，以便在计算机器 200 上正确执行。这种情况的例子可以是包括用于引用第二程序应用中资源的资源的文件管理程序应用，从而该文件管理程序应用依赖于第二程序应用中的资源的存在以便正确执行。当确定第一应用程序需要第二应用程序来正确执行时，将第一关联保存在存储器元件中（步骤 330），并且使第一应用程序在隔离环境 260 中启动（步骤 327）。方法 301 的其他实施例使得该应用程序在第三安装范围中启动，而另一些其他实施例使得该应用程序在用户范围中启动。

[0159] 在方法 301 的一个实施例中，如果确定第一应用程序的执行需要第二应用程序，则将所需要的第二应用程序的映像装到第二安装范围上（步骤 312）。方法 301 的其他实施例可以将应用程序装到第一安装范围。方法 301 的另一些其他实施例可以将应用程序的修改版本装到 (mount onto) 第二安装范围或者第一安装范围上。

[0160] 方法 301 的实施例随后建立第二应用和第二安装范围之间的第二关联（步骤

315)。在此实施例中,还建立第一安装范围和第二安装范围之间的关联(步骤318)。在方法301的一些实施例中,当第二应用程序装到第二安装范围时,产生第二应用程序和第二安装范围之间的关联(步骤312)。方法301的其它实施例包括在单独的安装范围中建立第一安装范围和第二安装范围之间的关联。方法301的另一些其他实施例通过将每个范围聚集到单个安装范围中而建立第一安装范围和第二安装范围之间的关联,而其他实施例通过进行以下任一操作来建立第一安装范围和第二安装范围之间的关联:建立表示第一安装范围和第二安装范围之间的关联的虚拟对象,在表、数据库、文件或者其它数据跟踪对象中产生表示该关联的条目;或者将标识符增加到这两个安装范围中任一以指示第一安装范围和第二安装范围的每一个应该互相关联。

[0161] 在方法301的一个实施例中,在第三安装范围和第一安装范围与第二安装范围的聚集之间建立第三关联(步骤321)。方法301的其他实施例建立包括第一安装范围和第二安装范围的第三安装范围。另一些其他实施例包括第三安装范围,其建立第三安装范围、第一安装范围和第二安装范围之间的共同关联,而不需要聚集第一安装范围和第二安装范围的每一个。

[0162] 保存第一关联、第二关联和第三关联(步骤324),并且使第一应用程序在隔离环境260中启动(步骤327)。在方法301的其他实施例中,每一个所建立的关联保存在存储器元件中,其中可以在其间保存超过三个的关联。另一些其他实施例包括的关联被"保存",这里所述的保持是将条目保存在表、列表或者其它数据跟踪对象中,其中每个条目为所建立的关联建立目录。

[0163] 虽然图7中示出的实施例所说明的方法301仅涉及第一和第二应用程序以及第一和第二安装范围,但是方法301的其他实施例可以涉及超过两个的应用程序和超过两个的安装范围。另一些其他实施例可以涉及X个应用程序和Y个安装范围,其中X是表示大于2的整数,而Y表示大于2且不同于X所表示数值的整数。

[0164] 详细的虚拟化示例

[0165] 上述方法和设备可以用来虚拟化更宽范围的原生资源258,以下描述多个这样的例子。

[0166] 文件系统虚拟化

[0167] 上述方法和设备可以被用来虚拟化对文件系统的访问。如上所述,文件系统通常组织为目录的逻辑分层,其自身就是文件并且可以包含其它目录和数据文件。

[0168] 总的来说,图8描述用来打开上述虚拟化环境中的文件所采取的步骤的一个实施例。接收或者拦截用来打开文件的请求(步骤602)。该请求包含隔离环境260认为是虚拟文件名称的文件名称。确定可应用于该文件系统打开请求的目标的处理规则(步骤604)。如果该规则动作是"重定向"(步骤642),则根据该应用的规则将在请求中提供的虚拟文件名称映射成文本文件名称(步骤606)。将使用文本文件名称打开文本文件的请求传递到操作系统210并且将源自操作系统210的结果返回给请求者(步骤640)。如果相反该规则动作是"忽略"(步骤642),则文本文件名称被确定为就是虚拟文件名称(步骤608),并且打开文本文件的请求传递给操作系统210,并且源自操作系统210的结果返回给请求者(步骤640)。如果步骤642中的规则动作是"隔离",则用户空间中对应于虚拟文件名称的文件名称被识别为候选文件名称(步骤610)。换句话说,通过将虚拟文件名称映射到可应用的用

户范围所专用的原生文件名称来形成候选文件名称。通过检查用户范围和与候选文件相关联的任一元数据，确定候选文件存在的类别 (catagory)（步骤 612)。如果因为用户范围中的该候选文件或者该候选文件的祖先目录的任一个被标记为删除而确定候选文件“否定存在”，这意味着所请求的虚拟文件已知不存在。在这种情况下，将指示所请求的文件未找到的错误状况返回给请求者（步骤 616)。如果相反在步骤 612 中的候选文件被确定为“肯定存在”，则因为候选文件存在于用户范围中并且候选文件未标以占位符节点，所以所请求的虚拟文件已知存在。候选文件被识别为针对该请求的文本文件（步骤 620)，并且发布请求以打开该文本文件并且将结果返回给请求者（步骤 630)。然而，如果在步骤 612，由于候选文件不存在或者候选文件存在但被标为占位符节点而候选文件具有“中性存在（无确定存在)”，则还不知道虚拟文件是否存在。在此情况中，调用子例程用来获取对应于虚拟文件名称的安装范围文件名称，将找到的安装范围文件名称识别为候选文件名称，并且通过检查安装范围和与所识别候选文件相关联的任一元数据来对所识别的候选文件的存在进行类别（步骤 614)。在一个实施例中，所调用的子例程是图 12 中描述的子例程的实施例。该子例程搜索隔离环境 260 中包括的一个或者多个安装范围来确定在一个或者多个安装范围中是否存在候选文件名称。在子例程的一个实施例中，由该子例程返回的值指示在一个或多个安装范围的任一中是否存在候选文件。如果从子例程返回的值指示候选文件具有“否定存在”，这是因为候选文件或者安装范围中的该候选文件的祖先目录之一被标以删除，则这意味着所请求的虚拟文件已经知道并不存在。在此情况中，将指示所请求文件没找到的错误状况返回给请求者（步骤 616)。如果相反在步骤 614 中从子例程返回的值指示候选文件被确定为具有“肯定存在”，这是因为候选文件存在于至少一个安装范围中并且未标以占位符节点，则已经知道存在所请求的虚拟文件。检查该请求来确定打开请求是否指示其目的意于修改该文件（步骤 622)。如果否，候选文件被识别为用于该请求的文本文件（步骤 620)，并且发布打开该文本文件的请求并且将结果返回给请求者（步骤 630)。然而，如果在步骤 622 中，确定打开请求指示其目的意于修改该文件，则检查和该文件相关联的许可数据来确定是否允许修改该文件（步骤 636)。如果否，将指示不允许修改该文件的错误状况返回给请求者（步骤 638)。如果许可数据指示该文件可以被修改，则将候选文件复制到用户范围（步骤 634)。在一些实施例中，候选文件被复制到规则引擎限定的位置。例如，规则可以指定文件被复制到安装范围。在其他实施例中，规则可以指定文件应该复制到的特定的安装子范围或者用户子范围。为了将复制的实例正确定位到分层中，将文件所复制到的范围中没有出现的所请求文件的任一祖先建立为该范围中的占位符 (placeholder)。该范围内的实例被识别为文本文件（步骤 632) 并且发布用来打开文件的请求并且将结果返回给请求者（步骤 630)。返回步骤 614，如果候选文件具有中性存在，这是因为候选文件不存在或者因为候选文件虽然被找到但标以占位符节点，则还不知道虚拟文件是否存在。在此情况中，将对应于虚拟文件名称的系统范围文件名称识别为候选文件名称（步骤 628)。换句话说，候选文件名称正是虚拟文件名称。如果候选文件不存在（步骤 626)，则将指示虚拟文件未找到的错误状况返回给请求者（步骤 624)。如果另一方面候选文件存在（步骤 626)，则检查该请求以确定打开请求是否指示其意于修改该文件（步骤 622)。如果否，将候选文件识别为对于该请求的文本文件（步骤 620)，并且发布用来打开文件的请求并且将结果返回给请求者（步骤 630)。然而，如果在步骤 622 中，确定该打开请求指示意于修改

该文件，则检查和该文件相关联的许可数据以确定是否允许修改该文件（步骤 636）。如果否，将指示不允许修改文件的错误状况返回给请求者（步骤 638）。如果许可数据指示该文件可以修改，将候选文件复制到用户范围（步骤 634）。在一些实施例中，候选文件被复制到规则引擎限定的位置。例如，规则可以指定文件被复制到安装范围。在其他实施例中，规则可以指定文件应该复制到的特定的安装子范围或者用户子范围。将范围中没有出现的所请求文件的任一祖先创建为范围中的占位符，以便将复制的实例正确定位到分层中。该范围内的实例被识别为文本文件（步骤 632）并且发布用来打开文件的请求并且将结果返回给请求者（步骤 630）。

[0169] 该实施例可以进行细微修改以执行对文件的存在的检查而非执行打开文件的检查。在步骤 630 中打开文本文件的尝试可以使用检查文本文件是否存在来代替，并且将该状态返回给请求者。

[0170] 继续参考图 8 并且更详细地，接收或者拦截用来打开虚拟文件的请求（步骤 602）。对应的文本文件可以在用户范围、安装范围或者系统范围中，或者其可以被划定范围到安装子范围或者用户子范围。在一些实施例中，通过替代用于打开文件的一个或多个操作系统 210 函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到进程建立时该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派针对原生文件的请求中使用的操作系统 210 资源相关联。对于为每一类文件操作提供单独的操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，单个钩子函数可以提供来拦截针对多种文件操作的建立或者打开调用。

[0171] 该请求包含被隔离环境 260 认为是虚拟文件名称的文件名称。通过查阅规则引擎，确定适用于文件系统打开请求的处理规则（步骤 604）。在一些实施例中，使用包括在打开请求中的虚拟名称来确定适用于该打开请求的处理规则。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，为所请求的文件提供的虚拟文件名称被用作规则引擎中的索引，来定位应用到该请求的一个或者多个规则。在这些实施例的特定例子中，对于一个特定文件可以在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟文件名称匹配的规则是应用到该请求的规则。在其他实施例中，如果存在可应用的规则，则使用进程标识符来在规则引擎中定位可应用到该请求的规则。和请求相关联的规则将忽略该请求，重定向该请求，或者隔离该请求。尽管图 6 所示为单个数据库事务处理（transaction）或者对文件的单次查找，但是规则查找也可以由一系列规则查找来实现。

[0172] 如果该规则动作是“重定向”（步骤 642），则根据可应用规则将在请求中提供的虚拟文件名称映射成文本文件名称（步骤 606）。打开由文本文件名称识别出的文本文件的请求被传递到操作系统 210 并且源自操作系统 210 的结果返回给请求者（步骤 640）。例如，打开名称为“file_1”的文件的请求可以导致打开名称为“Different_file_1”的文本文件。在一个实施例中，这是通过调用的被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，使用虚拟名称打开文件的第一请求导致从文件系统过滤器驱动返回 STATUS-REPARSE（状态重新解析）响应，其指示所确定的文本名称。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的

所确定的文本名称的文件打开请求。

[0173] 如果相反规则动作是“忽略”(步骤642),则文本文件名称被确定为就是虚拟文件名称(步骤608),并且打开文本文件的请求被传递给操作系统210,并且源自操作系统210的结果返回给请求者(步骤640)。例如,请求打开名称为“file_1”的文件导致打开实际名称为“file_1”的文件。在一个实施例中,这是通过调用被钩挂函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现。

[0174] 如果相反步骤642中的规则动作是“隔离”,则将对应于虚拟文件名称的用户范围文件名称识别为候选文件名称(步骤610)。换句话说,通过将虚拟文件名称映射到该可应用的用户范围所专用的原生文件名称来形成候选文件名称。例如,请求打开名称为“file_1”的文件导致打开实际名称为“Isolate_file_1”的文件。在一个实施例中,这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例,使用虚拟名称打开文件的第一请求导致从文件系统过滤器驱动返回STATUS-REPARSE响应,其指示所确定的文本名称。I/O管理器随后重新发布具有包括在STATUS-REPARSE响应中的所确定的文本名称的文件打开请求。

[0175] 在一些实施例中,为了隔离所请求的系统文件而形成的文本名称可以基于所接收的虚拟文件名称和范围专用的标识符。范围专用的标识符可以是和安装范围、用户范围、会话范围、安装子范围、用户子范围或者上述一些的组合相关联的标识符。范围专用的标识符可以用来“打乱(mangle)”请求中接收到的虚拟名称。

[0176] 在其他实施例中,用户范围或子范围可以是用户范围中所有文件所保存的目录。在一些这样的实施例中,用户目录下的目录树结构反应所请求资源的路径。换句话说,通过将虚拟文件路径映射到用户范围来形成文本文件路径。例如,如果所请求的文件是c:\temp\test.txt并且用户范围目录是d:\user1\app1\,,则到用户范围文本文件的路径可以是d:\user1\app1\c\temp\test.txt。在其他实施例中,到用户范围文本文件的路径可以以原生命名约定来限定。例如,到用户范围文本文件的路径可以是d:\user1\app1\device\harddisk1\temp\test.txt。在另一些其他实施例中,用户范围文件可以全部保存在具有选择为唯一名称的单个目录中并且可以使用数据库来保存所请求文件名和保存在目录中的对应文本文件的名称之间的映射。在另一些其他实施例中,文本文件的内容可以保存在数据库中。在另一些其他实施例中,原生文件系统为单个文件提供工具以包括多个独立命名的“流”,并且用户范围的文件的内容被保存为系统范围中的相关文件的附加流。此外,文本文件可以保存在可以设计来优化磁盘使用或者其他感兴趣的标准的自定义文件系统。

[0177] 通过检查用户范围和与候选文件相关联的任一元数据,确定候选文件的存在的类别(步骤612)。如果确定候选文件具有“否定存在”,这是因为该候选文件或者用户范围中的该候选文件的祖先目录之一被标以删除,则这意味着所请求的虚拟文件已经得知不存在。在此情况中,将指示所请求文件未找到的错误状况返回给请求者(步骤616)。

[0178] 在一些实施例中,关于文件的少量元数据可以直接保存在文本文件名中,诸如通过在虚拟名称后附加元数据指示符,其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在,通过虚拟文件名访问文件的请求需要检查文本文件名可能的变化,并且用于获取文件自身名称的请求被钩挂或者拦截,以便使用该文本名称做出响应。在其他实施例中,对于该文件

的一个或者多个替代名称可以根据虚拟文件名称和元数据指示符来形成，并且可以使用文件系统所提供的硬链接或者软链接工具来建立。隔离环境 260 可以通过在所给出的请求使用链接名称访问文件时指示没有找到该文件而对于各应用程序隐藏这些链接的存在。特定链接的存在或者缺失对于每一元数据指示符可以指示一位元数据，或者存在具有可以呈现多个状态来指示元数据多个位的元数据指示符的链接。仍在其他实施例中，在文件系统支持交替文件流 (alternate file stream) 的情况下，可以建立交替文件流来包含元数据，其中该流的尺寸指示多个元数据位。仍在其他实施例中，文件系统可以直接提供为文件系统中每一文件保存一些第三方元数据的能力。

[0179] 在特定的一个这样的实施例中，可以维持和查阅删除文件或者文件系统元件的列表，来优化对于删除文件的检查。在这些实施例中，如果重新建立所删除的文件，则文件名称可以从删除文件的列表中移除。在其他的这样的实施例中，如果该列表变得大于特定尺寸则将文件名从该列表中移除。

[0180] 如果相反候选文件被确定具有“肯定存在”，这是因为候选文件存在于用户范围中并且候选文件未标以占位符节点，则所请求的虚拟文件已经得知存在。候选文件被识别为对于该请求的文本文件（步骤 620），并且发布用以打开该文本文件的请求并且将结果返回给请求者（步骤 630）。

[0181] 然而，如果候选文件具有“中性存在”，这是由于候选文件不存在或者候选文件存在但被标为占位符节点，则还不知道虚拟文件是否存在。在此情况中，调用子例程来搜索一个或者多个安装范围以获取安装范围文件名称，并将该虚拟文件名称识别为候选文件名称（步骤 614）。换句话说，通过将虚拟文件名称映射到对应的原生文件名称来形成候选文件名称，其中该原生文件名称所专用于的安装范围被识别为其中可以找到一个对应的原生文件的安装范围。在一个实施例中，一旦子例程识别出候选文件，则子例程随后可以通过检查安装范围和任一元数据来确定候选文件存在的类别（步骤 614）。在一个实施例中，子例程返回用于指示候选文件存在的类别的值（步骤 614）。

[0182] 在一个实施例中，如果从子例程返回的值指示安装范围候选文件具有“否定存在”，并且该“否定存在”是候选文件或者该安装范围中的其祖先目录被标以删除的结果，则这意味着所请求的虚拟文件已经知道并不存在。在此实施例中，将指示所请求文件没找到的错误状况返回给请求者（步骤 616）。

[0183] 如果在步骤 614 中从子例程返回的值指示候选文件具有“肯定存在”，并且候选文件存在于安装范围中并且未标以占位符节点，则已经知道存在所请求的虚拟文件。检查该请求来确定该打开请求是否指示其目的意于修改该文件（步骤 622）。如果否，候选文件被识别为用于该请求的文本文件（步骤 620），并且发布用以打开该文本文件的请求并且将结果返回给请求者（步骤 630）。

[0184] 然而，如果在步骤 622 中确定打开请求指示其目的意于修改该文件，则检查和该文件相关联的许可数据来确定是否允许修改该文件（步骤 636）。在一些实施例中，许可数据和安装范围候选文件相关联，在一些这样的实施例中，许可数据保存在规则引擎或者和候选文件相关联的元数据中。在其他实施例中，和候选文件关联的许可数据由操作系统 210 提供。此外，规则引擎可以包括指示隔离环境 260 遵循或者无视对于资源的虚拟备份的原生许可数据的配置设置。在一些实施例中，该规则可以为一些虚拟资源指定修改将发生的

范围，例如系统范围、安装范围或子范围，或者用户范围或子范围。在一些实施例中，规则引擎可以基于所访问资源的类型或者分层来指定应用到虚拟化的原生资源的子集的配置设置。在一些这样的实施例中，配置设置可以专用于每个原生资源。在另一个例子中，规则引擎可以包括禁止或者允许修改特定文件类的配置数据，诸如可执行代码，MIME 类型或者如操作系统 210 限定的文件类型。

[0185] 如果和候选文件关联的许可数据指示不可以修改，则将指示不允许修改该文件的错误状况返回给请求者（步骤 638）。如果许可数据指示该文件可以被修改，则将候选文件复制到用户范围（步骤 634）。在一些实施例中，候选文件被复制到规则引擎所限定的位置。例如，规则可以指定文件被复制到另一个安装范围。在其他实施例中，规则可以指定文件应该复制到的特定的安装子范围或者用户子范围。没有出现在将文件复制到的范围中的所请求文件的任一祖先被建立为该范围中的占位符，以便将复制的实例正确定位到分层中。

[0186] 在一些实施例中，元数据和复制到该范围的文件相关联，用于识别该文件被复制的日期和时间。可以使用该信息来将和所复制的文件实例相关联的时间戳与位于较低隔离范围中的文件的另一个实例的时间戳或者对最初文件实例的最后一次修改的时间戳进行比较。在这些实施例中，如果文件的最初实例或者位于较低范围中的文件实例和一个晚于复制文件的时间戳的时间戳相关联，则该文件可以复制到该范围以更新该候选文件。在其他实施例中，范围中的文件的备份可以和识别包含被复制的最初文件的范围的元数据相关联。

[0187] 在进一步的实施例中，可以监控复制到范围的该文件来确定它们是否被实际修改，该文件被复制是由于它们已经出于修改目的而被打开。在一个实施例中，复制的文件可以在文件被实际修改时与一个置位的标志相关联。在这些实施例中，如果复制的文件事实上没有修改，其可以从在其关闭之后从其所复制到的范围中移除，以及删除和该所复制文件相关联的任一占位符节点。范围的实例被识别为文本文件（步骤 632）并且发布用来打开文件的请求并且将结果返回给请求者（步骤 630）。

[0188] 返回步骤 614。如果候选文件具有中性存在，这是由于候选文件不存在或者因为候选文件虽然找到但标以占位符节点，则还不知道虚拟文件是否存在。在此情况中，对应于虚拟文件名称的系统范围文件名称被识别为候选文件名称（步骤 628）。换句话说，候选文件名称正是虚拟文件名称。

[0189] 如果候选文件不存在（步骤 626），则将指示虚拟文件未找到的错误状况返回给请求者（步骤 624）。如果另一方面候选文件存在（步骤 626），则检查请求以确定打开请求是否指示其意于修改该文件（步骤 622）。

[0190] 如上，如果打开候选文件的目的不是对其修改，则将系统范围候选文件识别为对于该请求的文本文件（步骤 620），并且发布用来打开文件的请求并且将结果返回给请求者（步骤 630）。然而，如果在步骤 622 中确定打开请求指示意于修改该文件，则检查和该文件相关联的许可数据以确定是否允许修改该文件（步骤 636）。在一些实施例中，许可数据和系统范围候选文件相关联，在一些这样的实施例中，许可数据保存在规则引擎或者和候选文件相关联的元数据中。在其他实施例中，和候选文件关联的许可数据由操作系统 210 提供。

[0191] 如果和该系统范围候选文件相关联的许可数据指示不可以修改该文件，则将指示

不允许修改文件的错误状况返回给请求者（步骤638）。然而，如果许可数据指示该文件可以修改，候选文件被复制到用户范围（步骤634）。在一些实施例中，候选文件复制到由规则引擎限定的位置。例如，规则可以指定文件被复制到安装范围或者留在系统范围。在其他实施例中，规则可以指定文件应该复制到的特定的安装子范围或者隔离子范围。将没有出现在该范围中的所请求文件的任一祖先建立为范围中的占位符，以便将复制的实例正确定位到分层中。

[0192] 在一些实施例中，元数据和复制到该范围的文件相关联，用于识别出文件被复制的日期和时间。可以使用该信息来将和所复制文件实例相关联的时间戳和对最初文件实例的上一次修改的时间戳进行比较。在这些实施例中，如果文件的最初实例和一个晚于复制文件的时间戳的时间戳相关联，则最初文件可以复制到该范围以更新候选文件。在其他实施例中，复制到范围中的候选文件可以和用于识别从中复制出该最初文件的范围的元数据相关联。

[0193] 在进一步的实施例中，可以监控复制到范围的该文件来确定它们是否被实际修改，复制该文件是由于它们已经出于修改目的而被打开。在一个实施例中，复制的文件可以和在文件被实际修改时置位的标志相关联。在这些实施例中，如果复制的文件事实上没有修改，则其关闭之后可以从其复制到的范围中移除，以及移除和所复制文件相关联的任一占位符节点。仍在其他实施例中，在文件被实际修改时，文件仅复制到合适的范围。

[0194] 划定范围的实例被识别为文本文件（步骤632）并且发布用来打开文件的请求并且将结果返回给请求者（步骤630）。

[0195] 现在参考图9，总的来说，描述用来删除文件所采取的步骤的一个实施例。接收或者拦截用来删除文件的请求（步骤652）。该请求包含被隔离环境260认为是虚拟文件名称的文件名称。规则确定如何处理该文件操作（步骤654）。如果该规则动作是“重定向”（步骤656），则根据规则将虚拟文件名称映射到文本文件名称（步骤658）。删除文本文件的请求被传递到操作系统210并且将源自操作系统210的结果返回给请求者（步骤662）。如果规则动作是“忽略”（步骤656），则文本文件名称被识别为就是该虚拟文件名称（步骤660），并且将删除文本文件的请求传递给操作系统210，并且源自操作系统210的结果返回给请求者（步骤662）。如果规则动作是“隔离”（步骤656），则确定虚拟文件的存在（步骤664）。如果虚拟文件不存在，将将指示虚拟文件不存在的错误状况返回给请求者（步骤665）。如果虚拟文件存在，并且如果虚拟文件指定不同于一般文件的目录，则查询该虚拟目录以确定其是否包括任一虚拟文件或者虚拟子目录（步骤668）。如果所请求的虚拟文件是包含任一虚拟文件或者虚拟子目录的虚拟目录，则该虚拟目录不能被删除则返回错误消息（步骤670）。如果所请求的虚拟文件是一般文件或者是不包含虚拟文件和虚拟子目录的虚拟目录，则识别出对应于该虚拟文件的文本文件（步骤672）。检查和该文件相关联的许可数据以确定是否允许删除（步骤676）。如果否，则返回许可错误消息（步骤678）。然而，如果允许删除该文件，并且该虚拟文件处于合适的用户范围（步骤674），则删除该文本文件（步骤680）并且在合适的用户范围中建立表示所删除的虚拟文件的“删除”节点（步骤684）。然而，如果在步骤674中确定文本文件不处于用户范围中而处于合适的安装范围或者系统范围中，则建立所请求文件的用户范围的实例的用户范围祖先的还不存在的实例并且标以占位符（步骤686）。这样做是为了维持用户范围中的目录结构的逻辑分层。随后

在合适的用户范围中建立表示所删除的虚拟文件的用户范围的“删除”节点（步骤 684）。

[0196] 继续参考图 9 并且更详细地，接收或者拦截用来删除文件的请求（步骤 652）。文件可以处于用户范围、安装范围或者系统范围，或者一些可应用的子范围。在一些实施例中，通过用于替代用来删除文件的一个或多个操作系统 210 函数的函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到建立进程时的该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派对于原生文件的请求中使用的操作系统 210 资源相关联。对于为每一类文件提供单独的操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，单个钩子函数可以提供用来拦截对于多种类型的文件的建立或者打开调用。

[0197] 该请求包含被隔离环境 260 认为是虚拟文件名称的文件名称。通过查阅规则引擎，确定适用于删除操作的处理规则（步骤 654）。在一些实施例中，使用为所请求文件提供的虚拟文件名称来定位规则引擎中能应用于该请求的规则。在这些实施例的特定一个中，对于一个特定文件可能在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟文件名称匹配的规则是应用到该请求的规则。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，在该请求中提供的虚拟文件名称被用作规则引擎中的索引，来定位应用到该请求的一个或者多个规则。在其他实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则，如果该规则存在的话。和请求相关联的规则将忽略该请求，重定向该请求，或者隔离该请求。尽管图 9 所示为一系列决策，但是规则查找可以实现为单个数据库事务处理。

[0198] 如果该规则动作是“重定向”（步骤 656），则根据可应用规则将虚拟文件名称直接映射到文本文件名称（步骤 658）。将删除文本文件的请求传递到操作系统 210 并且将源自操作系统 210 的结果返回给请求者（步骤 662）。例如，删除名称为“file_1”的文件的请求可以导致删除名称为“Different_file_1”的实际文件。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，使用虚拟名称删除文件的第一请求导致从文件系统过滤器驱动返回 STATUS-REPARSE 响应，其指示所确定的文本名称。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的所确定的文本名称的文件删除请求。

[0199] 在一些实施例中，和文本文件“Different_file_1”相关联的操作系统 210 许可可以禁止文本文件的删除。在这些实施例中，返回文件不能删除的错误消息。

[0200] 如果规则动作是“忽略”（步骤 656），则文本文件名称被识别为正是虚拟文件名称（步骤 660），并且删除文本文件的请求被传递给操作系统，并且源自操作系统 210 的结果返回给请求者（步骤 662）。例如，请求删除名称为“file_1”的文件导致删除实际名称为“file_1”的文件。在一个实施例中，这是通过调用被钩挂函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，使用虚拟名称删除文件的第一请求导致从文件系统过滤器驱动返回 STATUS-REPARSE 响应，其指示该文本名称。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的所确定的文本名称的文件删除请求。

[0201] 在一些实施例中，和文本文件“file_1”相关联的操作系统 210 许可可以禁止文本文件的删除。在这些实施例中，返回文件不能删除的错误消息。如果规则动作是“隔离”（步骤 656），则确定虚拟文件的存在（步骤 664）。如果虚拟文件不存在，返回指示文件未找到的错误（步骤 665）。然而，如果在步骤 668 中确定文件存在并且文件不同于一般文件而且不是空的虚拟目录，即包含虚拟文件或者虚拟子目录，则返回指示该文件不能删除的错误消息（步骤 670）。

[0202] 然而，如果确定该文件存在并且所请求的虚拟文件是一般文件或者是不包含虚拟文件和虚拟子目录的空的虚拟目录（步骤 668），则识别出对应于该虚拟文件的文本文件（步骤 672）。按照隔离规则所指定的，根据虚拟文件名称确定出文本文件名称。例如，请求删除名称为“file_1”的文件导致删除名称为“Isolated_file_1”的实际文件。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，使用虚拟名称删除文件的第一请求导致从文件系统过滤器驱动返回 STATUS-REPARSE 响应，其指示文本名称。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的所确定的文本名称的文件删除请求。

[0203] 一旦识别出对应于虚拟文件的文本文件，则确定是否可以删除文本文件（步骤 676）。如果不可以删除该文件，则返回用于指示不可以删除文件的错误（步骤 678）。在一些实施例中，许可数据和系统范围候选文件相关联，在一些这样的实施例中，许可数据保存在规则引擎或者和候选文件相关联的元数据中。在其他实施例中，和候选文件关联的许可数据由操作系统 210 提供。然而，如果允许删除该文件，并且该文本文件处于合适的用户范围（步骤 674），则删除该文本文件（步骤 680）并且在合适的用户范围中建立表示所删除的虚拟文件的“删除”节点（步骤 684）。然而，如果在步骤 674 中确定文本文件不处于用户范围中而处于合适的安装范围或者系统范围中，则建立所请求文件的用户范围的实例的用户范围祖先的还不存在的实例并且标以占位符（步骤 686）。这样做是为了维持用户范围中的目录结构的逻辑分层。随后在合适的用户范围中建立表示所以删除的虚拟文件的用户范围的“删除”节点（步骤 684）。在一些实施例中，在文件或者其他高速缓存存储器中保存删除文件的标识，以优化对删除的文件的检查。

[0204] 在一些实施例中，所定位的虚拟文件可以和指示虚拟文件已经删除的元数据相关联，在一些其他实施例中，虚拟文件的祖先（例如，包含该文件的高级目录）和指示其被删除的元数据相关联。在这些实施例中，可以返回用于指示虚拟化文件不存在的错误消息。在这些实施例中的特定一个中，删除的文件或者文件系统元件的列表可以被维持和查询以优化对所删除文件的检查。

[0205] 现在参考图 10，示出在所描述的虚拟环境中列举目录所采用的步骤的实施例。接收或者拦截列举请求（步骤 760）。该请求包含被隔离环境 260 认为是虚拟目录名称的目录名称。原理上，如上述确定虚拟目录的存在（步骤 762）。如果虚拟目录不存在，将指示未找到虚拟目录的结果返回给请求者（步骤 796）。如果相反虚拟目录存在，则查阅规则引擎以确定用于列举请求中所指定的目录的规则（步骤 764）。如果该规则指定动作为“重定向”（步骤 766），则按照规则所指定的那样确定对应于虚拟目录名称的文本目录名称（步骤 768）并且列举由该文本名称所识别的文本目录，并且将列举结果保存在工作数据存储中（步骤 792），之后是如下描述的步骤 790。如果指定的规则动作不是“重定向”而是“忽

略”(步骤772),则文本目录名称正是虚拟目录名称(步骤770)并且列举文本目录,将列举结果保存在工作的数据存储中(步骤792),之后是如下描述的步骤790。然而,如果规则动作指定“隔离”,首先列举系统范围,也就是说,候选目录名称正是虚拟目录名称,并且如果候选目录存在,则将其列举。列举结果保存在工作数据存储中。如果候选目录不存在,则工作数据存储在此阶段保持空(步骤774)。接下来,执行子例程以识别隔离范围中的范围候选(步骤776)。子例程返回该候选(步骤776),并且候选目录被识别为虚拟目录的安装范围实例,并且确定候选目录的存在的分类(步骤778)。如果候选目录具有“否定存在”,即该候选目录或者该范围中的其祖先之一被标以删除,则在该范围内已经得知被删除,并且通过刷新工作数据存储将其指示(步骤780)。如果相反候选目录不具有否定存在,则列举候选目录并且将所得到的任一列举结果合并到工作数据存储中。特别地,对于列举中的每一个文件系统元素,确定存在的分类。将具有否定存在的元件从工作数据存储中移除并且将具有肯定存在的元件增加到工作数据存储中,替换在工作数据存储中已经存在的对应的元件,该肯定存在是指该元件存在并且未标为占位符并且未标以删除(步骤782)。

[0206] 在每一情况中,将候选目录识别为虚拟目录的用户范围实例,并且确定候选目录的存在的类别(步骤784)。如果候选目录具有“否定存在”,即该候选目录或者该范围中的该候选目录的一个祖先被标以删除,则该范围中已经知道被删除,并且这通过刷新工作数据存储加以指示(步骤786)。如果相反候选目录不具有否定存在,则列举候选目录并且将所得到的任一列举结果合并到工作数据存储中。特别地,对于列举中的每一个文件系统元件,确定其存在的分类。具有否定存在的元件从工作数据存储中移除并且具有肯定存在的元件被增加到工作数据存储中,替换在工作数据存储中已经存在的对应的元件,其中该肯定存在是指该元件存在并且未标为占位符并且未标以删除(步骤788),之后是如下描述的步骤790。

[0207] 之后,对于规则的全部三种类型,执行步骤790。查询规则引擎以找到其过滤器匹配所请求目录的直系子目录但不匹配所请求目录自身的规则集(步骤790)。对于该集中的每个规则,使用上述逻辑查询其名称与规则中的名称相匹配的虚拟子目录的存在。如果子目录具有肯定存在,则将其增加到工作数据存储,替代那里已经存在的相同名称的任一子目录。如果该子目录具有否定存在,则将对应于该子目录的工作数据存储中的条目移除,如果该条目存在的话(步骤794)。最后,从工作数据存储返回所构建的列举给请求者(步骤796)。

[0208] 继续参考图10并且更详细地,接收或者拦截列举目录的请求(步骤760)。在一些实施例中,通过用于替代一个或多个列举目录的操作系统函数的函数来钩挂该请求。在另一个实施例中,使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例,钩子函数可以加载到建立进程时该进程的地址空间。对于钩子函数在内核模式中执行的实施例,钩子函数可以和在分派对于文件操作的请求中使用的操作系统210资源相关联。对于为每一类文件操作提供单独操作系统210函数的实施例,每个函数可以被单独地钩挂。或者,单个钩子函数可以提供来拦截对于多种类型的文件操作的建立或者打开调用。

[0209] 确定虚拟目录的存在(步骤762)。如上所述将其实现。如果虚拟目录不存在,其不能被列举,并且将指示虚拟目录不存在的结果返回给请求者(步骤796)。

[0210] 该请求包含被隔离环境 260 认为是虚拟目录名称的目录名称。如果虚拟目录存在，则通过查阅规则引擎定位用于确定如何处理列举操作的规则（步骤 764）。在一些实施例中，规则引擎可以提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，为所请求的目录的虚拟目录名称被用于在规则引擎中定位应用到该请求的规则。在这些实施例的特定一个中，对于一个特定目录可能在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟目录名称相匹配的规则是应用到该请求的规则。在其他实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则，如果该规则存在的话。和请求相关联的规则可以忽略该请求，重定向该请求，或者隔离该请求。尽管图 10 所示为单个数据库事务处理或者对文件的单次查找，但是规则查找可以实现为一系列规则查找。

[0211] 如果该规则动作为“重定向”（步骤 766），则根据规则将虚拟目录名称直接映射到文本目录名称（步骤 768）。将列举文本目录的请求传递给操作系统 210（步骤 792），之后是如下描述的步骤 790。例如，列举名称为“directory_1”的目录的请求可以导致列举名称为“Different_directory_1”的文本目录。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，使用虚拟名称请求打开目录以进行列举的第一请求导致用于指示所确定的文本名称的“STATUS-REPARSE”请求响应。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的所确定的文本名称的针对列举的目录打开请求。

[0212] 如果规则动作不是“重定向”（步骤 776）而是“忽略”（步骤 772），则文本目录名称被识别为正是虚拟目录名称（步骤 770）并且列举文本目录的请求被传递给操作系统 210（步骤 792），之后是如下描述的步骤 790。例如，请求列举名称为“directory_1”的目录导致列举名称为“directory_1”的实际目录。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。对于使用文件系统过滤器驱动的实施例，将使用虚拟名称请求列举目录的第一请求传递通过该过滤器驱动而不作修改。

[0213] 如果步骤 772 中确定的规则动作不是“忽略”而是“隔离”，则列举系统范围，也就是说，该请求中提供的虚拟名称用来识别所列举的目录（步骤 774）。列举结果保存在工作数据存储中。在一些实施例中，工作数据存储包括存储器元件。在其他实施例中，工作数据存储包括数据库、文件、固态存储器元件或者永久数据存储。

[0214] 接下来，调用子例程来检查安装范围以识别并返回候选目录（步骤 776）。随后确定候选目录的存在（步骤 778）。如果候选目录具有“否定存在”，即该候选目录或者该范围中的该候选目录的祖先之一被标以删除，则该范围内已经得知被删除，并且这通过刷新工作数据存储加以指示（步骤 780）。

[0215] 在一些实施例中，关于文件的少量元数据可以直接保存在文本文件名中，诸如通过在虚拟名称之后添加元数据指示符，其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在，通过虚拟文件名访问文件的请求检查文本文件名可能的改变，并且获取文件自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中，文件的一个或者多个替代名称可以根据虚拟文件名称和元数据指示符来形成，并且可以使用文件系统所提供的

硬链接或者软链接工具来建立。隔离环境可通过在所给出的请求使用链接名称访问文件时指示没有找到该文件而对于各个应用程序隐藏这些链接。特定链接的存在或者缺失对于每一元数据指示符可以指示一位元数据，或者存在具有可以呈现多个状态来指示元数据多个位的元数据指示符的链接。仍在其他实施例中，其中文件系统支持交替文件流，可以建立交替代文件流来包含元数据，其中该流的尺寸指示多个元数据位。仍在其他实施例中，文件系统可以直接提供为文件系统中每一文件保存一些第三方元数据的能力。在又一些其它实施例中，可以使用单独的子范围来记录删除的文件，且文件在该子范围中的存在（未标为占位符）表示文件被删除。

[0216] 如果相反候选范围目录不具有否定存在，则列举候选目录并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个文件系统元件，确定其存在的分类。具有否定存在的元件被从工作数据存储中移除并且具有肯定存在的元件被增加到工作数据存储中，替换在工作数据存储中已经存在的对应的元件，该肯定存在是指该元件存在并且未标为占位符并且未标以删除（步骤 782）。

[0217] 在每一情况中，将候选目录识别为虚拟目录的用户范围实例，并且确定候选目录的存在的类别（步骤 784）。如果候选目录具有“否定存在”，即该候选目录或者该范围中该候选目录的一个祖先被标以删除，则已知其已经在该范围中被删除，并且这通过刷新工作数据存储加以指示（步骤 786）。如果相反候选目录不具有否定存在，则列举候选目录并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个文件系统元件，确定其存在的分类。具有否定存在的元件被从工作数据存储中移除并且具有肯定存在的元件被增加到工作数据存储中，替换在工作数据存储中已经存在的对应的元件，该肯定存在是指该元件存在并且未标为占位符并且未标以删除（步骤 788），之后是如下描述的步骤 790。

[0218] 之后，对于规则的全部三种类型，执行步骤 790。查询规则引擎以找到一规则集，该规则的过滤器与所请求目录的直接子目录，但不匹配所请求目录自身（步骤 790）。对于该规则集中的每个规则，使用上述逻辑查询其名称与规则中的名称相匹配的虚拟子目录的存在。如果该子目录具有肯定存在，则将其增加到工作数据存储，替代那里已经存在相同名称的任一子目录。如果该子目录具有否定存在，则移除对应于该子目录的工作数据存储中的条目，如果该条目存在的话（步骤 794）。最后，从工作数据存储返回所构建的列举给请求者（步骤 796）。

[0219] 本领域内的普通技术人员可以认识到上述分层的列举过程可以通过微小改变而应用于列举包括多个子范围的单个范围的的操作。建立工作数据存储，列举连续的子范围并且将结果合并到工作数据存储以形成范围的聚集列举。

[0220] 现在参考图 11 并且总的来说，示出隔离环境 260 中建立文件所采取的步骤的一个实施例。接收或者拦截用来建立文件的请求（步骤 702）。该请求包含被隔离环境 260 认为是虚拟文件名称的文件名称。使用完全虚拟化、利用可应用的规则来尝试打开所请求的文件，即如上所述使用合适的用户和安装范围（步骤 704）。如果访问被拒绝（步骤 706），则将访问拒绝错误返回给请求者（步骤 709）。如果访问被许可（步骤 706），并且所请求的文件被成功打开（步骤 710），则将所请求的文件返回给请求者（步骤 712）。然而，如果访问被许可（步骤 706），而所请求的文件没有成功打开（步骤 710），则如果所请求文件的父

文件也不存在（步骤 714），则将适合请求语义的错误发布给请求者（步骤 716）。如果另一方面，在使用合适的用户和安装范围的整个虚拟视图中找到所请求文件的父文件（步骤 714），该规则随后确定如何处理该文件操作（步骤 718）。如果规则动作是“重定向”或者“忽略”（步骤 720），则根据规则将虚拟文件名称直接映射给文本文件名称。特别地，如果规则动作是“忽略”，将文本文件名称识别为正是虚拟文件名称。如果相反规则动作是“重定向”，则按照规则所指定的那样根据虚拟文件名称确定文本文件名称。随后将建立文本文件的请求传递给操作系统 210，并将结果返回给请求者（步骤 724）。如果另一个方面，步骤 720 中确定的规则动作是“隔离”，则文本文件名称被识别为用户范围中的虚拟文件名称的实例。如果文本文件已经存在，而和指示其是占位符或者已经删除的元数据相关联，则修改所关联的元数据以移除那些指示，并且确保该文件是空的。在每一种情况中，打开文本文件的请求传递给操作系统 210（步骤 726）。如果文本文件被成功打开（步骤 728），则文本文件返回给请求者（步骤 730）。如果另一方面，在步骤 728 中，所请求的文件不能打开，对于当前在用户范围中并不存在的文本文件的每个祖先建立占位符（步骤 732），并且使用文本名称建立文本文件的请求被传递给操作系统 210 并且将结果返回给请求者（步骤 734）。

[0221] 继续参考图 11 并且更详细地，接收或者拦截用来建立文件的请求（步骤 702）。在一些实施例中，由替代用于建立文件的一个或多个操作系统 210 函数的函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以加载到建立进程时该进程的地址空间中。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派对于文件的请求中使用的操作系统 210 资源相关联。对于为每一类文件操作提供单独操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，单个钩子函数可以提供来拦截对于多种文件操作的建立或者打开调用。

[0222] 该请求包含被隔离环境 260 认为是虚拟文件名称的文件名称。请求者试图使用可应用的规则（即如上所述使用合适的用户和安装范围）、通过完全的虚拟化来打开所请求的文件（步骤 704）。如果在完全的虚拟化打开操作期间访问被拒绝（步骤 706），则将访问拒绝错误返回给请求者（步骤 709）。如果访问被许可（步骤 706），并且所请求的虚拟文件被成功打开（步骤 710），则将对应的文本文件返回给请求者（步骤 712）。然而，如果访问被许可（步骤 706），而所请求的文件没有成功打开（步骤 710），则虚拟文件已经被确定不存在。如果如上述过程所确定的那样所请求的虚拟文件的虚拟父文件也不存在（步骤 714），则将适合请求语义的错误发布给请求者（步骤 716）。如果另一方面，在使用合适的用户和安装范围的整个虚拟视图中找到所请求虚拟文件的虚拟父文件（步骤 714），则随后通过查阅规则引擎定位用于确定如何处理该建立操作的规则（步骤 718）。在一些实施例中，规则引擎可以提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，为所请求的文件提供的虚拟文件名称被用于在规则引擎中定位一个应用到该请求的规则。在这些实施例的特定一个中，对于特定文件可以在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟文件名称相匹配的规则是应用到该请求的规则。在其他实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则，如果该规则存在的话。和请求相关联的规则将忽略该请求，重定向该请求，或者隔离该请求。尽管图 11 所示为单个数据库事务处理或者对文件的单次查找，但是规则

查找可以实现为一系列规则查找。

[0223] 如果规则动作是“重定向”或者“忽略”(步骤 720),则根据规则将虚拟文件名称直接映射给文本文件名称(步骤 724)。如果规则动作是“重定向”(步骤 720),则按照规则所指定的那样根据虚拟文件名称确定文本文件名称(步骤 724)。如果规则动作是“忽略”(步骤 720),则确定文本文件名称正是虚拟文件名称(步骤 724)。如果规则动作是“忽略”或者规则动作是“重定向”,随后将使用所确定的文本文件名称的用于建立文本文件的请求传递给操作系统 210,并将来自操作系统 210 的结果返回给请求者(步骤 724)。例如,建立名称为“file_1”的文件的请求可以导致建立名称为“Different_file_1”的文本文件。在一个实施例中,这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的(步骤 724)。对于使用文件系统过滤器驱动的实施例,使用虚拟名称打开文件的第一请求导致用于指示所确定的文本名称的 STATUS-REPARSE 请求响应。I/O 管理器随后重新发布具有包括在 STATUS-REPARSE 响应中的所确定的文本名称的文件打开请求。

[0224] 如果步骤 720 中确定的规则动作不是“忽略”或者“重定向”而是“隔离”,则文本文件名称被识别为用户范围中虚拟文件名称的实例。如果文本文件已经存在,而和指示其是占位符或者已经删除的元数据相关联,则修改所关联的元数据以移除那些指示,并且确保该文件是空的。

[0225] 在一些实施例中,关于文件的少量元数据可以直接保存在文本文件名中,诸如通过在虚拟名称之后添加元数据指示符,其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在,通过虚拟文件名访问文件的请求检查文本文件名的可能变化,并且获取文件自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中,对于文件的一个或者多个替代名称可以根据虚拟文件名称和元数据指示符来形成,并且可以使用文件系统所提供的硬链接或者软链接工具来建立。隔离环境 260 可通过在所给出的请求使用链接名称访问文件时指示没有找到该文件而对于各个应用程序隐藏这些链接的存在。特定链接的存在或者缺失对于每一元数据指示符可以指示一位元数据,或者存在具有可以呈现多个状态来指示元数据多个位的元数据指示符的链接。仍在其他实施例中,其中文件系统支持交替文件流,可以建立交替文件流来包含元数据,其中流的尺寸指示多个元数据位。仍在其他实施例中,文件系统可以直接提供为文件系统中的每一文件保存一些第三方元数据的能力。

[0226] 在特定的一个这样的实施例中,可以维持和查询删除的文件或者文件系统元件的列表,以优化对于删除文件的检查。在这些实施例中,如果重新建立所删除的文件,则文件名称可以从删除的文件的列表中移除。在其他的这样的实施例中,当列表变得超出特定尺寸时将文件名称从该列表中移除。

[0227] 在每一种情况中,打开用户范围文本文件的请求被传递给操作系统 210(步骤 726)。在一些实施例中,规则可以指定对应于虚拟文件的文本文件应该在除用户范围之外的范围中建立,诸如安装范围、系统范围、用户子范围或者安装子范围。

[0228] 如果文本文件被成功打开(步骤 728),则文本文件被返回给请求者(步骤 730)。如果另一方面,在步骤 728 中,所请求的文件不能打开,则为在用户范围内当前不存在文本文件的每个祖先建立占位符(步骤 732),并且使用文本名称建立文本文件的请求被传递给

操作系统 210 并且将结果返回给请求者（步骤 734）。该实施例用于具有每次调用仅支持一级建立的 API 或者工具的操作系统 210。延伸到每次调用进行多级建立的情况对于本领域内的普通技术人员来说是显而易见的。

[0229] 在一些文件系统中，每一个文件可以给予短的或者长的文件名。每一种名称都可以用来在上述文件操作的任一个中访问文件。对于拥有短和长两种文件名的每个文件，这隐含地在分配给该文件的短文件名和长文件名之间建立关联。在一些这样的文件系统中，短文件名可以由文件系统自动分配给使用长文件名建立的文件。如果隔离环境 260 没有维持短和长文件名之间的关联，在同一目录中但在不同范围级中具有不同长文件名的文件可以具有相同的短文件名，这导致在使用短文件名访问虚拟文件时的混淆。或者，当文件复制到用户范围以进行修改时可以改变该短文件名，这意味着虚拟文件不再能使用原始的短文件名加以访问。

[0230] 为了防止产生这些问题，首先用于将以修改为目的而打开的文件实例复制到更高范围的文件系统操作保存和所复制实例相关联的短文件名和长文件名之间的关联。其次，对于新建立的隔离文件建立唯一的短文件名，以代替操作系统所分配的文件名。所产生的短文件名应该满足以下条件，即所产生的文件名不与同一范围中同一目录下或者较低范围中同一目录下的任一已有的短文件名相匹配。例如，为用户范围中的文件实例产生的短文件名不应该匹配该目录中的安装范围实例中或者该目录中的系统范围实例中已有的短文件名。

[0231] 现在参考图 11A，示出在建立新文件之后分配唯一短文件名所采取步骤的一个实施例。总的来说，进行检查以确定是否应该产生短文件名（步骤 752）。如果否，返回指示不产生短文件名的状态（步骤 758）。否则，检查文件名来确定其是否是根据文件系统的合法的短文件名（步骤 754）。如果已经是合法的短文件名，返回指示不产生短文件名的状态（步骤 756）。否则，构建合适的短文件名（步骤 756）。

[0232] 继续参考图 11A 并且更详细地，进行检查以确定是否应该产生短文件名（步骤 752）。在一些实施例中，基于用于保存该文件名所引用的文件的装置来做出这一决策。在其他实施例中，对于特定范围或者子范围或者对于作为一个整体的隔离环境 260，可以启用短文件名的产生。在一些这样的实施例中，注册表设置可以指定对于特定文件名是否产生短文件名。如果短文件名不应该产生，则返回不产生短文件名的状态（步骤 758）。

[0233] 否则，检查文件名来确定其是否其已经是合法的短文件名（步骤 754）。在一些实施例中，合法的短文件名包含最多八个字符的文件名和最多三个字符的可选扩展名。在一些实施例中，合法短文件名仅包含合法字符，诸如 A-Z、a-z、0-9、‘、～、！、@、#、$、％、Λ、&、*、(,)、-、、′、{,and}。在一些实施例中，最前面的空格或者“.”或包含超过一个“.”都是不合法的。如果所提供的文件名已经是合法的短文件名，则返回不产生短文件名的状态（步骤 758）。

[0234] 否则，如果在步骤 754 中确定文件名是不合法的短文件名，则构建合适的短文件名（步骤 756）。在一些实施例中，这是通过使用一些对于在短文件名中使用依然合法的部分长文件名与编码后的重复计数相结合以形成候选短文件名来实现的。增加该重复计数，直到所关联的候选短文件名适合，也就是其是在同一范围的同一目录中或者较低范围的同一目录中的任意其他文件未使用的合法短文件名。在其他实施例中，长文件名被打乱或者

哈希并且编码，以及和编码后的重复计数相结合以形成候选短文件名。增加该重复计数，直到所关联的候选短文件名适合，也就是其是在同一范围的同一目录中或者较低范围的同一目录中的任意其他文件未使用的合法短文件名。在所有这些实施例中，范围专用的串可以被并入到候选短文件名中，以增加找到具有低重复计数的合适的候选短文件名的可能性。

[0235] 图 12 中示出在图 8 所示过程中调用的子例程 614 的一个实施例。该子例程是图 8 中所述方法 601 中执行的方法，用来在包括在隔离环境 260 中的一个或者多个安装范围上循环。当图 8 中的方法 601 的步骤 612 中确定用户范围候选文件的存在具有中性存在时，调用子例程 614 并且在第一安装范围中识别候选文件（步骤 1724）。随后确定在第一安装范围中是否真正找到一个候选文件（步骤 1702）。如果找到候选文件，则检查该找到的候选文件是否具有否定存在，该否定存在指示该找到的文件已经标记为删除或者以其他方式不再出现在该安装范围中（步骤 1704）。当候选具有否定存在，进行检查以确定在隔离环境 260 中是否存在多个安装范围（步骤 1710），然而，如果候选文件不具有否定存在，则进行检查以确定候选文件是否具有中性存在（步骤 1706）。当候选文件具有中性存在，这指示文件可以是占位符或者该文件可以不存在，并且由此进一步分析隔离范围 260 以确定隔离环境 260 中是否存在附加的安装范围（步骤 1710）。如果候选文件不具有中性存在，则将指示候选文件具有肯定存在的值返回给主方法 601（步骤 1708）。如果在步骤 1724 之后，确定候选文件没有找到（步骤 1702），则分析隔离环境 260 以确定在隔离环境 260 中是否存在附加的安装范围（步骤 1710）。当确定存在附加的安装范围时，在下一个安装范围中识别出第二候选文件（步骤 1712）。一旦分析下一个安装范围，则确定在该安装范围中是否找到候选文件（步骤 1714）。当找到候选文件时，随后检查该文件以确定是否具有否定存在（步骤 1704），或者中性存在（步骤 1706）。如果没有找到第二候选文件，则检查是否存在附加的安装范围（步骤 1710）。当识别出没有更多的附加安装范围时（步骤 1710）时，检查在上一次检查期间是否找到候选文件（步骤 1724，步骤 1712），并且如果没有找到，则将否定值返回给主方法 601（步骤 1722）。如果候选文件被找到（步骤 1716），则确定找到的候选文件是否具有中性存在（步骤 1718）。当候选文件具有中性存在时，将中性值返回给主方法 601（步骤 1720），并且当候选文件不具有中性存在时，将否定值返回给主方法 601（步骤 1722）。

[0236] 继续参考图 12 并且更详细地，在子例程 614 的一个实施例中，在第一安装范围中识别候选文件（步骤 1724）。其他实施例可以包括在最后一个安装范围或者安装范围的位置是用户限定值或者预定值的另一个安装范围中检查候选文件的子例程 614。另一些其他实施例包括子例程 614，其中，第一安装范围是隔离环境 260 中最高优先级安装范围的安装范围。子例程 614 的实施例可以包括作为隔离环境 260 中最低优先级安装范围的第一安装范围，第一安装范围在安装范围的子范围中具有最高优先级，或者第一安装范围在安装范围的子范围中具有最低优先级。随后确定在第一安装范围中是否找到候选文件（步骤 1702）。在过程 614 的一个实施例中，当在第一安装范围中找到候选文件，则确定该文件是否具有否定存在（步骤 1704）、中性存在（步骤 1706）或者肯定存在（步骤 1708）。当该文件具有否定存在，进一步确定是否存在附加的安装范围（步骤 1710），如果不存在其他安装范围，则验证找到候选文件（步骤 1716），并且确定候选文件是否具有中性存在（步骤 1718）或者否定存在（步骤 1722）。当文件不具有否定存在，确定该文件是否具有中性存在（步骤 1706），并且如果候选文件不具有中性存在（步骤 1706），则将指示候选文件具有肯定存

在的肯定值返回给主方法 601（步骤 1708）。如果文件具有中性存在，则检查附加的安装范围是否包括在隔离环境 260 中（步骤 1710），如果不存在其他安装范围，则验证候选文件找到（步骤 1716）并且确定候选文件是否是中性（步骤 1718）或者否定的（步骤 1722）。

[0237] 在一个实施例中，当在第一安装范围中没有找到候选文件，确定在隔离环境 260 中是否存在附加的安装范围（步骤 1710）。其他实施例可以被配置以搜索预定数量的安装范围。当存在附加的安装范围，搜索下一个安装范围以识别候选文件（步骤 1712）。在一个实施例中，下一个安装范围是优先级低于之前检查的安装范围的优先级的安装范围。例如，在此实施例中，如果之前检查的安装范围是具有最高优先级的第一安装范围，则下一个安装范围可以是具有次高优先级的安装范围。当不存在附加的安装范围时，确定该候选文件是否是之前识别出的（步骤 1716）。在一个实施例中，之前识别的候选文件包括在检查附加的安装范围的存在之前由方法 614 识别出的文件。

[0238] 当在隔离环境 260 中包括附加的安装范围时，该方法试图识别每一安装范围中的候选文件（1712）。在一个实施例中，确定是否找到候选文件（步骤 1714），并且如果找到候选文件，则分析该文件以确定该文件具有否定存在（步骤 1704）、中性存在（步骤 1706）或者肯定存在（步骤 1708）。当没有找到候选文件（1714），则检查用于多个安装范围（步骤 1710），并且如果不存在附加安装范围，则验证找到候选文件（步骤 1716）。如果没找到候选文件，则返回否定值（步骤 1722），并且如果找到候选文件，则确定候选文件是否具有中性（步骤 1718）。

[0239] 图 13 中所示为图 10 中的方法 798 所调用的子例程 776 的一个实施例。在一个实施例中，通过图 10 中的过程 798 调用子例程 776，以在包括在隔离环境 260 中的所有安装范围上循环执行，并且识别具有肯定存在的安装范围。在将列举的系统范围结果保存在工作数据存储之后（步骤 774），识别第一安装范围的候选者（步骤 1742）。确定第一安装范围的候选者是否具有否定存在（步骤 1744）。当第一安装范围具有肯定存在时，将肯定值返回给主过程 798（步骤 1746），并且安装范围被列举到工作数据存储中（步骤 782）。当第一安装范围具有否定存在时，检查隔离环境中附加安装范围的存在（步骤 1748）。如果不存在附加的安装范围，则将否定存在返回给主过程 798（步骤 1754）。如果存在附加的安装范围，则识别下一个候选安装范围（步骤 1750）并且确定下一安装范围是否具有否定存在（步骤 1744）。验证下一个候选安装范围。

[0240] 继续参考图 13 并且更详细地，当从主过程 798 调用子例程 776 时，在图 10 中所示的实施例中，识别第一安装范围的候选者（步骤 1742）。在一个实施例中，第一安装范围的候选者是比包括在隔离环境 260 中的所有其他安装范围具有更高优先级的安装范围。其他实施例所包括的第一安装范围是比包括在隔离环境 260 中所有其他安装范围具有更低优先级的安装范围。仍旧其他实施例可以包括具有比其他安装子范围更高优先级的第一安装范围，或者具有比包括在子隔离环境 260 中的其他安装范围更高优先级的第一安装范围。

[0241] 过程 776 的实施例确定安装范围是否具有否定存在（步骤 1744）。在一个实施例中，当该安装范围具有否定存在时（步骤 1744），检查隔离环境 260 是否包括附加安装范围（步骤 1748）。在另一个实施例中，当安装范围不具有否定存在时（步骤 1744），将肯定值返回给主过程 798（步骤 1746）。

[0242] 方法 776 的一个实施例确定隔离环境 260 中是否存在附加的安装范围（步骤

1748)。在一个实施例中，当不存在附加的安装范围时，则将否定值返回给主过程 798(步骤 1746)，而在其它实施例中，当在隔离环境 260 中存在附加的安装范围时，识别下一个候选安装范围(步骤 1750)。

[0243] 注册表虚拟化

[0244] 上述方法和设备可以被用来虚拟化对注册表数据库的访问。如上所述，注册表数据库保存关于以下内容的信息：物理接附到计算机 200 的硬件，哪个系统选项已被选定，计算机存储器如何建立，各种应用程序专用数据项，在操作系统 210 启动时应该出现的应用程序。注册表数据库 220 通常组织成键值的逻辑分层结构，其中键值为注册表值的容器。

[0245] 总的来说，图 14 描述用来打开上述隔离环境 260 中的注册表键值所采取的步骤的一个实施例。接收或者拦截用来打开注册表键值的请求(步骤 802)。该请求包含被隔离环境 260 认为是虚拟键值名称的注册表键值名称。适用于该请求中的虚拟名称的处理规则确定如何处理该注册表键值操作(步骤 804)。如果该规则动作是“重定向”(步骤 806)，则按照可应用规则所指定的那样，将在该请求中提供的虚拟键值名称映射成文本键值名称(步骤 808)。使用文本键值名称打开文本注册表键值的请求被传递到操作系统 210 并且源自操作系统 210 的结果返回给请求者(步骤 810)。如果规则动作不是“重定向”，而是“忽略”(步骤 806)，则虚拟键值名称被识别为文本键值名称(步骤 812)，并且打开文本注册表键值的请求被传递给操作系统 210，并且源自操作系统 210 的结果返回给请求者(步骤 810)。如果步骤 806 中确定的规则动作不是“重定向”也不是“忽略”而是“隔离”，则将该请求中提供的虚拟键值名称映射成用户范围候选键值名称，其是专用于可应用的用户范围中的对应于虚拟键值名称的键值名称(步骤 814)。通过检查用户范围和与候选键值相关联的任一元数据，确定用户范围候选键值的存在的分类(步骤 816)。如果确定候选键值具有“否定存在”，这是因为候选键值或者用户范围中的其祖先键值之一被标以删除，这意味着所请求的虚拟键值已经得知不存在。在此情况中，将指示所请求文件未找到的错误状况返回给请求者(步骤 822)。如果相反在步骤 816 中的候选键值被确定具有“肯定存在”，这是因为该候选键值存在于用户范围中并且候选键值未标以占位符节点，所以所请求的虚拟键值已经得知存在。候选键值被识别为对于该请求的文本键值(步骤 818)，并且发布用于打开该文本键值的请求并且将结果返回给请求者(步骤 820)。然而，如果在步骤 816 中由于候选键值不存在或者候选键值存在但被标为占位符节点而候选注册表键值具有“中性存在”，则还不知道虚拟键值是否存在。在此情况中，调用子例程来将对应于虚拟键值名称的安装范围键值名称识别为候选键值名称，并且进一步验证该候选键值的存在的类别(步骤 824)。换言之，该子例程通过将虚拟键值名称映射到专用于可应用的安装范围的对应的原生键值名称来建立候选键值名称，并且通过检查安装范围和与该候选键值相关联的任一元数据来确定候选键值的存在的类别(步骤 824)。该子例程将指示候选键值的存在的类别的值返回给主方法 811。如果返回的值是“否定存在”，则候选键值或者安装范围中的其祖先键值被标以删除，则这意味着所请求的虚拟键值已经知道并不存在。在此情况中，将指示所请求键值没找到的错误状况返回给请求者(步骤 822)。如果相反在步骤 826 中候选键值被确定具有“肯定存在”，这是因为候选键值存在于安装范围中并且未标以占位符节点，则已经知道存在所请求的虚拟键值。检查该请求来确定该打开请求是否指示其目的意于修改该键值(步骤 828)。如果否，候选键值被识别为用于该请求的文本键值(步骤 818)，并且发

布打开该文本键值的请求并且将结果返回给请求者（步骤820）。然而，如果在步骤828中确定该打开请求指示其目的意于修改该键值，则检查和该键值相关联的许可数据来确定是否允许修改该键值（步骤836）。如果否，将指示不允许修改该键值的错误状况返回给请求者（步骤838）。如果许可数据指示该键值可以被修改，则将候选键值复制到用户范围（步骤840）。在一些实施例中，候选键值被复制到规则引擎所限定的位置。例如，规则可以指定该键值被复制到安装范围。在其他实施例中，规则可以指定该键值应该被复制到的特定的安装子范围或者用户子范围。将在该键值复制到的范围中没有出现的所请求键值的任一祖先建立为该范围中的占位符，以便将复制的实例正确定位到分层中。新近复制的划定范围的实例被识别为文本键值（步骤842）并且发布用来打开该文本键值的请求并且将结果返回给请求者（步骤820）。返回步骤826。如果候选键值具有中性存在，这是因为由于候选键值不存在或者因为候选键值虽然被找到但标以占位符节点，则还不知道虚拟键值是否存在。在此情况中，对应于虚拟键值名称的系统范围键值名称被识别为候选键值名称（步骤830）。换句话说，候选键值名称正是虚拟键值名称。如果候选键值不存在（步骤832），则将指示虚拟键值未找到的错误状况返回给请求者（步骤834）。如果另一方面候选键值存在（步骤832），则检查该请求以确定该打开请求是否指示其意于修改该键值（步骤828）。如果否，将候选键值被识别为对于该请求的文本键值（步骤818），并且发布用来打开文本键值的请求并且将结果返回给请求者（步骤820）。然而，如果在步骤828中，确定打开请求指示意于修改该键值，则检查和该键值相关联的许可数据以确定是否允许修改该键值（步骤836）。如果否，将指示不允许修改键值的错误状况返回给请求者（步骤838）。如果许可数据指示该键值可以修改，则将候选键值复制到用户范围（步骤840）。在一些实施例中，候选键值被复制到规则引擎所限定的位置。例如，规则可以指定键值被复制到安装范围。在其他实施例中，规则可以指定该键值应该被复制到的特定的安装子范围或者用户子范围。将范围中没有出现的所请求键值的任一祖先建立为该范围中的占位符，以便将复制的实例正确定位到分层中。新近复制的划定范围的实例被识别为文本键值（步骤842）并且发布用来打开该文本键值的请求并且将结果返回给请求者（步骤820）。

[0246] 继续参考图14并且更详细地，接收或者拦截用来打开虚拟注册表键值的请求（步骤802）。对应的文本注册表键值可以处于用户范围、安装范围或者系统范围，或者其可以将范围限定到安装子范围或者用户子范围。在一些实施例中，由替代操作系统210用于打开注册表键值的一个或多个函数的函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到建立进程时该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派对于原生注册表键值的请求中使用的操作系统资源相关联。对于为每一类注册表键值操作提供单独操作系统210函数的实施例，每个函数可以被单独地钩挂。或者，可以提供单个钩子函数来拦截对于多种注册表键值操作的建立或者打开调用。

[0247] 该请求包含被隔离环境260认为是虚拟注册表键值名称的注册表键值名称。通过查阅规则引擎，确定适用于注册表键值打开请求的处理规则（步骤804）。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，为所请求的注册表键值提供的虚拟注册表

键值名称被用于在规则引擎中定位应用到该请求的规则。在特定的一些这样的实施例中，对于一个特定注册表键值可以在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟注册表键值名称相匹配的规则是应用到该请求的规则。在其他实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则，如果该规则存在的话。和请求相关联的规则可能将忽略该请求，重定向该请求，或者隔离该请求。尽管图 14 所示为单个数据库事务处理或者在文件中的单次查找，但是规则查找也可以以一系列规则来实现。

[0248] 如果该规则动作是“重定向”（步骤 806），则根据该可应用的规则将在请求中提供的虚拟注册表键值名称映射成文本注册表键值名称（步骤 808）。使用文本注册表键值名称打开文本注册表键值的请求被传递到操作系统 210 并且源自操作系统 210 的结果返回给请求者（步骤 810）。例如，打开名称为“registry_key_1”的注册表键值的请求可以导致打开名称为“Different_registry_key_1”的文本注册表键值。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。在其他实施例中，操作系统 210 可以提供类似于文件系统过滤器驱动工具的注册表过滤器驱动工具。在这些实施例中，通过向注册表过滤器管理器发信号以便使用所确定的文本键值名称来重新解析该请求而响应于该打开虚拟键值的原始请求，可以实现打开文本注册表键值的操作。如果相反规则动作是“忽略”（步骤 806），则文本注册表键值名称被确定为正式虚拟注册表键值名称（步骤 812），并且打开文本注册表键值的请求被传递给操作系统，并且源自操作系统的结果返回给请求者（步骤 810）。例如，请求打开名称为“registry_key_1”的注册表键值导致打开实际名称为“registry_key_1”的文本注册表键值。在一个实施例中，通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。在另一个实施例中，这是通过向注册表过滤器管理器发信号以继续以正常方式处理最初未修改的请求来实现的。

[0249] 如果步骤 806 中的规则动作是“隔离”，则对应于虚拟注册表键值名称的用户范围注册表键值名称被识别为候选注册表键值名称（步骤 814）。换句话说，通过将虚拟注册表键值名称映射到可应用的用户范围专用的对应原生注册表键值名称来形成候选注册表键值名称。例如，请求打开名称为“registry_key_1”的注册表键值可以导致打开名称为“Isolated_UserScope_UserA_registry_key_1”的文本注册表键值。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现。在其它实施例中，通过向注册表过滤器管理器发信号以使用所确定的文本键值名称来重新解析该请求而响应于打开虚拟键值的原始请求，可以实现打开文本注册表键值。

[0250] 在一些实施例中，为了隔离所请求的虚拟注册表键值而形成的文本名称可以基于所接收的虚拟注册表键值名称和范围专用的标识符。范围专用的标识符可以是和安装范围、用户范围、会话范围、安装子范围、用户子范围或者上述一些组合相关联的标识符。范围专用的标识符可以用来“打乱”请求中接收到的虚拟名称。

[0251] 在其他实施例中，用户范围和子范围可以是用户范围中所有注册表键值都保存在其下的注册表键值。在一些这样的实施例中，用户范围键值下的键值分层反应所请求资源的路径。换句话说，通过将虚拟键值路径映射到用户范围来形成文本键值路径。例如，如果所请求的键值是 HKLM\Software\Citrix\MyKey，并且用户范围键值是 HKCU\Software\UserScope，则到用户范围文本键值的路径可以是 HKCU\Software\UserScope\HKLM\

Software\Citrix\MyKey。在其他实施例中，到用户范围文本的路径可以以原生命名约定来定义。例如，到用户范围文本键值的路径可以是：HKCU\Software\UserScope\Registry\Machine\Software\Citrix\MyKey。仍在其他实施例中，用户范围键值可以全部保存在具有选择为唯一名称的单个键值下并且可以使用数据库来保存所请求键值名称和保存在用户键值中的对应文本键值的名称之间的映射。仍在其他实施例中，文本键值的内容可以保存在数据库或者文件存储中。

[0252] 通过检查用户范围和与候选键值相关联的任一元数据，确定候选键值的存在的类别（步骤 816）。如果确定候选键值具有“否定存在”，这是因为该候选键值或者用户范围中的该候选键值的一个祖先键值被标以删除，这意味着所请求的虚拟键值已经得知不存在。在此情况中，将指示所请求键值未找到的错误状况返回给请求者（步骤 822）。

[0253] 在一些实施例中，文本注册表键值可以和指示虚拟注册表键值已经删除的元数据相关联。在一些实施例中，关于注册表键值的元数据可以保存在由该键值保持的特别的值中，其中该值的存在对于注册表 API 的一般应用程序使用而言是隐藏的。在一些实施例中，关于注册表键值的少量元数据可以直接保存在文本键值名称中，诸如通过在虚拟名称之后添加元数据指示符，其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在，通过虚拟名称访问键值的请求需检查文本键值名称的可能的变化，并且获取键值自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中，元数据指示符可以被编码在子键值名称或者注册表值名称中，而不是键值名称自身中。仍在其他实施例中，注册表键值系统可以直接提供为每一键值保存一些第三方元数据的能力。在一些实施例中，元数据保存在数据库或者与注册表数据库分离的其它知识库中。在一些实施例中，单独的子范围可以用来保存标以删除的键值。子范围中键值的存在指示该键值标以删除。

[0254] 在特定一个这样的实施例中，可以维持和查询一个删除的键值或者键值系统元件的列表，来优化对于删除键值的检查。在这些实施例中，如果重新建立删除的键值，则键值名称可以从该删除键值的列表中移除。在其他的这样的实施例中，在该列表变得超过特定尺寸时，从该列表中移除键值名称。

[0255] 如果相反在步骤 816 中候选键值被确定具有“肯定存在”，这是因为候选注键值存在于用户范围中并且未标以占位符节点，则所请求的虚拟键值已经得知存在。候选键值被识别为用于该请求的文本键值（步骤 818），并且发布用以打开文本键值的请求并且将结果返回给请求者（步骤 820）。

[0256] 然而，在步骤 816 中，如果候选键值具有“中性存在”，这是该候选键值不存在或者该候选键值存在但被标为占位符节点，则还不知道该虚拟键值是否存在。

[0257] 在此情况中，调用子例程来获取对应于虚拟键值名称的安装范围键值名称，将找到的安装范围键值名称识别为候选键值名称，并且通过检查安装范围和与所识别出的候选键值相关联的任一元数据来确定候选键值的存在的类别（步骤 824）。在一个实施例中，所调用的子例程是图 18 中描述的子例程的实施例。该子例程搜索隔离环境中包括的一个或者多个安装范围以确定候选键值名称是否存在于一个或者多个安装范围中。换句话说，通过将虚拟键值名称映射到专用于可应用的安装范围的对应的原生键值名称，来形成候选键值名称。在一个实施例中，如通过检查安装范围和与候选键值相关联的任一元数据所确定

的那样，由子例程返回指示候选键值的存在的类别的值（步骤 824）。

[0258] 如果安装范围候选键值具有“否定存在”，这是因为该候选键值或者安装范围中的该候选键值的一个祖先键值被标以删除，这意味着所请求的虚拟键值已经知道并不存在。在此情况中，将指示所请求键值没找到的错误状况返回给请求者（步骤 822）。

[0259] 然而，如果在步骤 824 中候选键值被确定具有“肯定存在”，这是因为候选键值存在于安装范围中并且未标以占位符节点，则已经知道存在所请求的虚拟键值。检查该请求来确定该打开请求是否指示其目的意于修改该键值（步骤 828）。如果否，候选键值被识别为用于该请求的文本键值（步骤 818），并且发布打开该文本键值的请求并且将结果返回给请求者（步骤 820）。

[0260] 然而，如果在步骤 828 中，确定打开请求指示其目的意于修改该键值，则检查和该键值相关联的许可数据来确定是否允许修改该键值（步骤 836）。在一些实施例中，许可数据和安装范围候选键值相关联，在一些这样的实施例中，许可数据保存在规则引擎或者和候选键值相关联的元数据中。在其他实施例中，和候选键值关联的许可数据由操作系统 210 提供。此外，规则引擎可以包括指示隔离环境 260 遵循或者无视用于对资源的虚拟备份的原生许可数据的配置设置。在一些实施例中，该规则可以为一些虚拟资源指定修改将发生的范围，例如系统范围、安装范围或子范围，或者用户范围或子范围。在一些实施例中，规则引擎可以指定基于分层结构应用到虚拟化原生资源的子集的配置设置。在一些这样的实施例中，配置设置可以专用于每个原子原生资源。

[0261] 如果和候选键值关联的许可数据指示不可以修改该键值，则将指示不允许修改该键值的错误状况返回给请求者（步骤 838）。如果许可数据指示该键值可以被修改，则将候选键值复制到用户范围（步骤 840）。在一些实施例中，候选键值被复制到规则引擎限定的位置。例如，规则可以指定键值被复制到另一个安装范围。在其他实施例中，规则可以指定键值应该复制到的特定的安装子范围或者用户子范围。将该键值复制到的范围中没有出现的所请求键值的任一祖先建立为范围中的占位符，以便将复制的实例正确定位到分层中。

[0262] 在一些实施例中，元数据和复制到该范围的键值相关联，用于识别该键值被复制的日期和时间。该信息可以用来将和所复制的键值实例相关联的时间戳与位于较低隔离范围中的键值的另一个实例或者对该键值的最初实例进行上一次修改的时间戳进行比较。在这些实施例中，如果键值的最初实例或者较低范围中的键值实例和一个晚于复制键值的时间戳的时间戳相关联，则该键值可以被复制到该范围以更新候选键值。在其他实施例中，该范围中的键值的备份可以和用于识别包含有被复制的原始键值的范围的元数据相关联。

[0263] 在进一步的实施例中，可以监控因已经以修改目的打开而被复制到范围的键值来确定它们是否被实际修改。在一个实施例中，复制的键值可以和一个在该键值被实际修改时置位的标志相关联。在这些实施例中，如果复制的键值事实上没有修改，其可以在其关闭之后从所复制到的范围中移除，而且和所复制的键值相关联的任一占位符节点也移除。

[0264] 划定范围的实例被识别为文本键值（步骤 842）并且发布用来打开文本键值的请求并且将结果返回给请求者（步骤 820）。

[0265] 返回步骤 826，如果候选键值具有中性存在，这是因为候选键值不存在或者因为候选键值虽然被找到但标以占位符节点，则还不知道虚拟键值是否存在。在此情况中，对应于虚拟键值名称的系统范围键值名称被识别为候选键值名称（步骤 830）。换句话说，候选键

值名称正是虚拟键值名称。

[0266] 如果候选键值不存在（步骤832），则将指示虚拟键值未找到的错误状况返回给请求者（步骤834）。如果另一方面候选键值存在（步骤832），则检查该请求以确定该打开请求是否指示其意于修改该键值（步骤828）。

[0267] 如上，如果打开该候选键值的目的不是对其修改，则将系统范围候选键值识别为对于该请求的文本键值（步骤818），并且发布用来打开该文本键值的请求并且将结果返回给请求者（步骤820）。然而，如果在步骤828中确定该打开请求指示意于修改该键值，则检查和该键值相关联的许可数据以确定是否允许修改该键值（步骤836）。在一些实施例中，许可数据和系统范围候选键值相关联，在一些这样的实施例中，许可数据被保存在规则引擎或者和该候选键值相关联的元数据中。在其他实施例中，和该候选键值关联的许可数据由操作系统210提供。此外，规则引擎可以包括用于指示隔离环境260遵循或者无视用于虚拟化的资源备份的原生许可数据的配置设置。在一些实施例中，该规则可以为一些虚拟资源指定修改将发生的范围，例如系统范围、安装范围或子范围，或者用户范围或子范围。在一些实施例中，规则引擎可以基于分层指定可应用到虚拟化的原生资源的子集的配置设置。在一些这样的实施例中，配置设置可以专用于每个原子原生资源。

[0268] 如果和该系统范围候选键值相关联的许可数据指示不可以修改该键值，则将指示不允许修改键值的错误状况返回给请求者（步骤838）。然而，如果许可数据指示该键值可以修改，则将该候选键值复制到用户范围（步骤840）。在一些实施例中，候选键值被复制到由规则引擎限定的位置。例如，规则可以指定键值被复制到安装范围或者其可以留在系统范围。在其他实施例中，规则可以指定该键值应该被复制到的特定的安装子范围或者用户子范围。将该范围中没有出现的该所请求键值的任一祖先建立为该范围中的占位符，以便将所复制的实例正确定位到分层中。

[0269] 在一些实施例中，元数据和复制到该范围的键值相关联，用于识别键值被复制的日期和时间。该信息可以用来将和所复制的键值实例相关联的时间戳与对该键值的最初实例的上一次修改的时间戳进行比较。在这些实施例中，如果键值的最初实例和一个晚于所复制的键值的时间戳的时间戳相关联，则该最初键值可以被复制到该范围以更新候选键值。在其他实施例中，复制到该范围中的候选键值可以和用于识别该最初键值被从中复制出的范围的元数据相关联。

[0270] 在进一步的实施例中，键值因已经以对其进行修改为目的打开而被复制到范围，该键值可以被监控以确定它们实际上是否被修改。在一个实施例中，复制的键值可以和一个在该键值实际被修改时置位的标志相关联。在这些实施例中，如果复制的键值事实上没有修改，则该键值可以在其关闭之后从其被复制到的范围中移除，而且和该所复制键值相关联的任一占位符节点也被移除。仍在进一步的实施例中，在键值实际被修改时，该键值仅被复制到该合适的范围。

[0271] 范围的实例被识别为文本键值（步骤842）并且发布用来打开该文本键值的请求并且将结果返回给请求者（步骤820）。

[0272] 现在参考图15，并且总的来说，其描述了为了删除注册表键值所采用的步骤的实施例。在键值删除之前，首先以删除访问打开该键值（步骤901）。如果该键值没有成功打开，则返回错误（步骤916）。如果虚拟键值成功打开，则接收或者拦截删除虚拟注册表键

值的请求，该请求包括到与该虚拟键值相对应的文本键值的句柄（步骤 902）。规则确定如何处理该注册表键值操作（步骤 904）。除了应用到待删除的键值的规则之外，检查可应用于直接子键值的任一其它规则（步骤 905）。对于找到的可应用于直接子键值的每一规则，尝试打开一个虚拟子键值，该虚拟子键值的名称由在步骤 905 中找到的规则中给定的名称所指定。如果未能成功打开其名称与在步骤 905 中找到的其中一个规则相对应的子键值（步骤 906），则虚拟键值被认为具有多个子键值，其意味着不能被删除，并且返回错误（步骤 907）。

[0273] 如果在步骤 905 中提取的所有虚拟键值名称已经被试图打开之后（步骤 906）并且发现并不存在虚拟键值，则需要进一步检查。如果规则动作不是“隔离”而是“重定向”或者“忽略”（步骤 908），则将一个删除该文本键值的请求传递给操作系统 210 并且将来自该操作系统的结果返回到请求者（步骤 911）。然而，如果在步骤 908 中确定的规则动作是“隔离”，则查阅聚集的虚拟注册表键值来确定其是否包含任一虚拟子键值（步骤 914）。如果该虚拟键值具有虚拟子键值，则删除不能继续执行，并且返回指示该键值还没有删除的错误（步骤 920）。如果虚拟键值不具有虚拟子键值，则检查对应于该虚拟键值的文本键值以确定其是否使用在另一个范围级中的相同的虚拟名称来掩盖该划定范围的键值（步骤 922）。如果对应于该虚拟键值的文本键值没有使用相同的虚拟名称来掩盖一个不同范围的键值，则删除对应于该虚拟键值的文本键值并且返回结果（步骤 926）。如果对应于该虚拟键值的文本键值使用相同的虚拟名称来掩盖一个不同范围的键值，则对应于该虚拟键值的文本键值被标以一个指示其被删除的值，并且返回成功结果给调用者（步骤 924）。

[0274] 继续参考图 15，并且更详细地，为了删除键值，首先使用删除访问打开该键值（步骤 901）。以删除访问打开该键值的请求包括隔离环境 260 认为是虚拟名称的键值名称。如上所述执行整个虚拟化键值打开。如果该虚拟化打开操作失败，返回错误给请求者（步骤 916）。如果虚拟化打开操作成功，则将对应于虚拟键值的文本键值的句柄返回给请求者。随后接收或拦截用来删除在步骤 901 中打开的注册表键值的请求（步骤 902）。该打开的文本注册表键值可以属于用户范围、安装范围、系统范围或者一些可应用的子范围。在一些实施例中，由替代用于删除注册表键值的一个或多个操作系统 210 函数的函数来钩挂该删除请求。在另一个实施例中，使用钩子动态链接库来拦截该删除请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到建立进程时该进程的地址空间中。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派对原生注册表键值的请求中使用的操作系统 210 资源相关联。在其他实施例中，操作系统 210 可以提供类似于文件系统过滤器驱动的注册表过滤器驱动。本领域内的普通技术人员可以建立注册表过滤器驱动，操作系统 210 将请求传递给该驱动以执行注册表操作，从而提供了拦截注册表操作请求的机制。对于为每一类注册表键值操作提供单独的操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，可以提供单个钩子函数来拦截对于多种注册表键值功能的建立或者打开调用。

[0275] 该删除请求包含文本键值句柄。通过在操作系统 210 中查询与该句柄相关联的文本名称可以确定和该句柄相关联的虚拟键值名称。查阅规则引擎以确定和该文本名称相关联的虚拟名称（如果存在的话）。通过查阅该规则引擎可以得到用于确定如何处理该注册表键值操作的规则（步骤 904）。在一些实施例中，使用要删除的虚拟注册表键值的虚拟键

值名称来在规则引擎中定位一个可应用于该请求的规则。在这些实施例的特定一个中，对于特定虚拟注册表键值可能在规则引擎中存在多个规则，并且在一些这样的实施例中，最长前缀和虚拟键值名称匹配的规则是应用到该请求的规则。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面注册表键值数据库。在一些实施例中，对应于该请求中的虚拟键值句柄的虚拟键值名称被用作规则引擎中的索引，以定位应用到该请求的一个或多个规则。在其他实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则（如果其存在的话）。和请求相关联的规则可能忽略该请求，重定向该请求，或者隔离该请求。规则查找可以作为一系列决策发生，或者规则查找可以作为单个数据库事务处理发生。

[0276] 要删除的键值的虚拟名称被用来查阅规则引擎以定位一个规则集，该规则集可应用于要删除的虚拟键值的任一直系子键值，但不可应用于该要删除的虚拟键值。无论那些子键值是否存在，都定位该规则集（步骤905）。如果可应用于直系子键值的规则集非空，则提取这些规则的每一个中的虚拟名称。随后，尝试对所提取的虚拟子键值名称的每一个进行完整的虚拟化打开（步骤906）。如果对应于这些虚拟名称的任一的任一虚拟键值不能被成功打开，则这意味着虚拟子键值的存在。这意味着由于该虚拟键值具有已经存在的虚拟子女，该虚拟键值不能删除，并返回错误（步骤907）。如果在检查了可应用于该虚拟键值的直系子女的所有规则集之后（步骤905），没找到存在的虚拟子键值，则可以继续删除。例如，具有虚拟名称“key_1”的键值可以具有可应用于“key1\subkey_1”和“key1\subkey2”的子规则。在此步骤中，尝试对“key1\subkey_1”和“key1\subkey2”的进行虚拟化打开。如果这两个虚拟子键值的任一个都可以成功打开，则删除失败，并返回错误（步骤907）。如果这两个虚拟子键值都不存在，则继续执行删除。

[0277] 如果规则动作不是“隔离”而是“重定向”或者“忽略”（步骤908），则将使用文本键值句柄删除文本注册表键值的请求传递给操作系统210并且将来自操作系统的结果返回到请求者（步骤911）。如果文本键值包含文本子键值，则该请求失败。在一个实施例中，通过调用被钩挂的函数的原始版本并且将该文本键值句柄作为参量传递到该函数来实现该文本注册表键值的请求。在使用注册表过滤器驱动的实施例中，这是通过以发信号命令操作系统执行对该请求的正常处理的完成状态来响应该请求来实现的。在一些实施例中，和该文本注册表键值关联的操作系统许可可以禁止删除。在这些实施例中，返回虚拟注册表键值不能删除的错误消息。

[0278] 然而，如果在步骤908中确定的规则动作是“隔离”，则查阅聚集的虚拟注册表键值来确定其是否包含任一虚拟子键值（步骤914）。如果所请求的虚拟注册表键值具有虚拟子键值，则不能删除该虚拟键值，并且返回错误给调用者（步骤920）。

[0279] 如果所请求的虚拟注册表键值不包含虚拟子键值，则删除该虚拟键值。接下来采取的动作依赖于包含待删除的文本键值的范围。例如删除虚拟注册表键值的请求可以导致删除应用程序范围的文本键值。通过查阅具有到该文本键值的全路径的规则引擎可以确定包含该文本键值的范围。

[0280] 如果在特定范围中找到要删除的文本键值，并且该文本键值掩盖了另一个范围中具有相同的虚拟名称的另一个键值，则待删除的文本键值被标以删除，并且结果被返回给请求者（步骤924）。例如，如果具有相同虚拟名称的对应的应用程序范围的键值或者具

有相同虚拟名称的对应的系统范围键值具有“肯定存在”，即存在该范围中并且未标以占位符，并且不被认为删除，则对应于用户范围的文本键值的虚拟键值被认为要掩盖一个不同范围的键值。类似地，如果系统范围键值存在并且未被认为删除，则应用范围的键值被认为要掩盖与相同虚拟名称相对应的系统范围的键值。

[0281] 如果发现要删除的文本键值并不会掩盖另一范围中具有相同虚拟名称的另一个键值，则确实删除该要删除的该文本键值并且返回结果（步骤 926）。

[0282] 在一些实施例中，和文本注册表键值关联的操作系统许可可以禁止删除该文本注册表键值。在这些实施例中，返回该虚拟注册表键值不能删除的错误消息。

[0283] 在一些实施例中，文本注册表键值可以和用于指示该虚拟注册表键值已经删除的元数据相关联。在一些实施例中，关于注册表键值的元数据可以保存在由该键值保持的特别的值中，该值的存在对于注册表 API 的一般应用程序使用而言是隐藏的。在一些实施例中，关于注册表键值的少量元数据可以直接保存在文本键值名称中，诸如通过在虚拟名称之后添加元数据指示符，其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在，以虚拟名称访问键值的请求检查文本键值名称的各个可能的变化，并且用于获取键值自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中，元数据指示符可以被编码在子键值名称或者注册表值名称中，而不是键值名称自身中。仍在其他实施例中，注册表键值系统可以直接为每一键值提供保存一些第三方元数据的能力。在一些实施例中，元数据被保存在数据库或者与注册表数据库分离的其它知识库中。在一些实施例中，单独的子范围可以被用来保存标以删除的键值。该子范围中键值的存在指示该键值标以删除。

[0284] 在特定一个这样的实施例中，可以维持和查找一个删除的键值或者键值系统元件的列表，以优化对于删除的键值的检查。在这些实施例中，如果重新建立所删除的键值，则该键值名称可以从删除键值的列表中移除。在其他的这样的实施例中，在该列表变得超过特定尺寸时键值名称可以从该列表中移除。

[0285] 在一些实施例中，同一范围中文本注册表键值的祖先可以和指示其已被删除的元数据相关联，或者以另外方式被指示删除。在这些实施例中，可以返回用于指示虚拟注册表键值不存在的错误消息。在这些实施例的特定实例中，可以维持和查找一个删除的注册表键值或者注册表键值系统元件的列表，以优化对于删除键值的检查。

[0286] 现在参考图 16，并且总的来说，示出了在所描述的虚拟环境中列举键值的过程 1003 的实施例。在键值被列举之前，首先使用列举访问打开该键值（步骤 1001）。如果该键值没有被成功打开，则返回错误（步骤 1040）。如果虚拟键值被成功打开，则拦截或者接收列举请求，该请求包括到与虚拟键值相对应的文本键值的句柄（步骤 1002）。

[0287] 确定对应于该句柄的虚拟键值名称，并且查阅规则引擎以确定用于该列举请求中指定的键值的规则（步骤 1004）。如果该规则不是指定“隔离”动作，而是指定“忽略”或者“重定向”动作（步骤 1006），则列举由该文本键值句柄所识别出的文本键值，并且将列举结果保存在工作数据存储中（步骤 1012），之后是如下描述的步骤 1030。

[0288] 然而，如果规则动作指定了“隔离”，首先列举系统范围，也就是说，候选键值名称正是虚拟键值名称，并且如果候选键值存在，则将其列举。列举结果被保存在工作数据存储中。如果候选键值不存在，则工作数据存储在此阶段保持为空（步骤 1014）。接下来，调用

子例程来在各个安装范围上循环执行以识别出候选键值并且验证该候选键值存在的类别（步骤 1015）。在一些实施例中，所调用子例程是图 19 中示出的子例程。在其他实施例中，子例程返回一个表示候选键值存在的类别的值（步骤 1015）。如果候选键值具有“否定存在”，即，该候选键值或者在该范围中的该候选键值的一个祖先被标以删除，则在该范围内已经得知其被删除，并且这可以通过刷新工作数据存储来指示出（步骤 1042）。如果相反候选键值不具有否定存在，则列举候选键值并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个子键值，确定其存在的分类。将具有否定存在的子键值从工作数据存储中移除，并且将具有肯定存在的子键值增加到工作数据存储中，从而如果在工作数据存储中已经存在的对应的子键值则替代该子键值，该肯定存在是指该子键值存在并且未标为占位符并且未标以删除（步骤 1016）。

[0289] 在这两种情况任一中，将该候选键值识别为虚拟键值的用户范围实例，并且确定该候选键值存在的类别（步骤 1017）。如果候选键值具有“否定存在”，即该候选键值或者该范围中该候选键值的一个祖先被标以删除，则在该范围中已经知道其被删除，并且这通过刷新（flush）工作数据存储来指示（步骤 1044）。如果相反候选键值不具有否定存在，则列举候选键值并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个子键值，确定其存在的类别。将具有否定存在的子键值从工作数据存储中移除，并且将具有肯定存在的子键值增加到工作数据存储中，从而如果在工作数据存储中已经存在对应的子键值则替换该对应子键值，该肯定存在是指该子键值存在并且未标为占位符并且未标以删除（步骤 1018），之后是如下描述的步骤 1030。

[0290] 之后，对于规则的全部三种类型，执行步骤 1030。查询规则引擎以找到一个规则集，该规则集的过滤器匹配所请求的虚拟键值名称的直接子键值，但不匹配所请求的虚拟键值名称本身（步骤 1030）。对于该集中的每个规则，确定名称与规则中的名称相匹配的虚拟子键值的存在。如果该子键值具有肯定存在，则将其增加到工作数据存储，从而替代那里已经存在的相同名称的任一子键值。如果该子键值具有否定存在，将移除对应于该子键值的工作数据存储中的条目（如果其存在的话）（步骤 1032）。最后，从工作数据存储返回所构建的列举给请求者（步骤 1020）。

[0291] 继续参考图 16 并且更详细地，为了列举一个键值，首先使用列举访问打开该键值（步骤 1001）。使用列举访问打开键值的请求包括隔离环境 260 认为是虚拟名称的键值的名称。如上所述执行整个虚拟化键值打开。如果该虚拟化打开操作失败，返回错误给请求者（步骤 1040）。如果该虚拟化打开操作成功，则将对应于虚拟键值的文本键值的句柄返回给请求者。随后，拦截或者接收用于列举在步骤 1001 中打开的注册表键值的请求（步骤 1002）。该打开的文本注册表键值可以属于用户范围、安装范围、系统范围或者一些可应用的子范围。在一些实施例中，由替代用于列举注册表键值的一个或多个操作系统 210 函数的函数来钩挂该列举请求。在另一个实施例中，使用钩子动态链接库来拦截该列举请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到建立进程时该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派对于原生注册表键值的请求中使用的操作系统 210 资源相关联。在其他实施例中，操作系统 210 可以提供类似于文件系统过滤器驱动工具的注册表过滤器驱动工具。本领域内的普通技术人员可以建立注册表过滤器驱动，操作系统 210 将

请求传递给该驱动以执行注册表操作，因此提供了拦截注册表操作请求的机制。对于每一类注册表键值功能提供单独操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，可以提供单个钩子函数来拦截对于多种注册表键值功能的建立或者打开调用。

[0292] 该列举请求包含文本键值句柄。通过在操作系统 210 中查询和该句柄相关联的文本名称，可以确定和该句柄按关联的虚拟键值名称。查阅该规则引擎以确定和该文本名称相关联的虚拟名称（如果其存在的话）。

[0293] 通过查阅规则引擎可得到用于确定如何处理注册表键值操作的规则（步骤 1004）。在一些实施例中，使用要列举的虚拟注册表键值的虚拟键值名称来在规则引擎中定位一个可应用于该请求的规则。在这些实施例的特定实例中，对于一个特定的虚拟注册表键值可以在规则引擎中存在多个规则，并且在一些这样的实施例中，最长前缀和虚拟键值名称相匹配的规则是应用到该请求的规则。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平的注册表键值数据库。在一些实施例中，对应于请求中的虚拟键值句柄的虚拟键值名称被用作规则引擎中的索引，以定位应用到该请求的一个或多个规则。在一些实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则（如果其存在的话）。和请求相关联的规则可以忽略该请求，重定向该请求，或者隔离该请求。规则查找可以作为一系列决策发生，或者规则查找可以作为单个数据库事务处理发生。

[0294] 如果该规则动作不是“隔离”而是“忽略”或者“重定向”（步骤 1006），则使用文本键值句柄将列举文本键值的请求传递给操作系统 210，并且将列举结果（如果有的话）保存在工作数据存储中（步骤 1012），之后是如下描述的步骤 1030。

[0295] 在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。在其他实施例中，操作系统 210 可以提供类似于文件系统过滤器驱动工具的注册表过滤器驱动工具。在这些实施例中，通过发信号令注册表过滤器管理器以正常方式处理未修改的请求而对列举键值的原始请求做出相应来实现对文本注册表键值的列举。

[0296] 如果步骤 1010 中确定的规则动作是“隔离”，则列举系统范围，为了实现这一点，将候选键值识别为对应于待列举的虚拟键值的系统范围键值。列举候选键值，并且将列举结果保存在工作数据存储中（步骤 1014）。在一些实施例中，工作数据存储包括存储器元件。在其他实施例中，工作数据存储包括数据库、键值、固态存储器元件或者永久数据存储。

[0297] 接下来，调用子例程来在各个安装范围上循环执行，将候选键值识别为虚拟键值的安装范围实例，并且确定候选键值存在的类别（步骤 1015）。如果候选键值具有“否定存在”，即该候选键值或者该范围中的该候选键值的一个祖先被标以删除，则在该范围内已经得知其被删除，并且这通过刷新工作数据存储来指示（步骤 1042）。

[0298] 在一些实施例中，候选注册表键值可以和用于指示候选注册表键值已经被删除的元数据相关联。在一些实施例中，关于注册表键值的元数据可以保存在由该键值保持的特别的值中，该值的存在对于注册表 API 的一般应用程序使用而言是隐藏的。在一些实施例中，关于注册表键值的少量元数据可以直接保存在文本键值名称中，诸如通过在虚拟名称之后添加元数据指示符，其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在，以虚拟名称访

问键值的请求检查该文本键值名称各个可能的变化，并且用于获取该键值自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中，元数据指示符可以被编码在子键值名称或者注册表值名称中，而不是键值名称本身中。仍在其他实施例中，注册表键值系统可以直接为每一键值提供保存一些第三方元数据的能力。在一些实施例中，元数据被保存在数据库或者与注册表数据库分离的其它知识库中。在一些实施例中，单独的子范围可以被用来保存标以删除的键值。在该子范围中键值的存在指示该键值被标以删除。

[0299] 如果相反在步骤 1015 中候选键值不具有否定存在，则列举该候选键值并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个子键值，确定其存在的类别。将具有否定存在的子键值从工作数据存储中移除，并且将具有肯定存在的子键值增加到工作数据存储中，从而如果在工作数据存储中已经存在对应的子键值则替换该子键值，其中该肯定存在是指该子键值存在并且未标为占位符并且未标以删除（步骤 1016）。

[0300] 在这两种情况任一中，将候选键值识别为该虚拟键值的用户范围实例，并且确定该候选键值存在的类别（步骤 1017）。如果该候选键值具有“否定存在”，即该候选键值或者该范围中的该候选键值的一个祖先被标以删除，则该范围中已经知道其被删除，并且这通过刷新工作数据存储来指示（步骤 1044）。如果相反候选键值不具有否定存在，则列举该候选键值并且将所得到的任一列举结果合并到工作数据存储中。特别地，对于列举中的每一个子键值，确定其存在的类别。将具有否定存在的子键值从工作数据存储中移除并且将具有肯定存在的子键值增加到工作数据存储中，从而如果在工作数据存储中已经存在对应的子键值则替换该子键值，其中该肯定存在是指该子键值存在并且未标为占位符并且未标以删除（步骤 1018），之后是如下描述的步骤 1030。

[0301] 之后，对于规则的全部三种类型，执行步骤 1030。查询规则引擎以找到一个规则集，该规则集的过滤匹配所请求键值的直系子键值，但不匹配所请求键值自身（步骤 1030）。对于该规则集中的每个规则，确定名称与规则中的名称相匹配的虚拟子键值的存在。在一些实施例中，这是通过检查合适的范围以及和该虚拟子键值相关联的和元数据来确定的。在其他实施例中，这是通过尝试打开该键值来确定。如果打开请求成功，则该虚拟子键值具有肯定存在。如果打开请求失败，且指示该虚拟子键值不存在，则该虚拟子键值具有否定存在。

[0302] 如果该子键值具有肯定存在，则将其增加到工作数据存储，从而替代那里已经存在相同名称的任一子键值。如果该子键值具有否定存在，则将在工作数据存储中对应于该虚拟子键值的条目（如果其存在的话）移除（步骤 1032）。最后，从工作数据存储返回所构建的列举给请求者（步骤 1020）。

[0303] 本领域内的普通技术人员可以认识到上述分层的列举过程可以微小改变应用到于用来列举包括多个子范围的单个范围的操作。建立一个工作数据存储，列举连续的子范围并且将结果合并到该工作数据存储以形成聚集的范围列举。

[0304] 现在参考图 17 并且总的来说，示出隔离环境 260 中建立键值所采取的步骤的一个实施例。接收或者拦截用来建立键值的请求（步骤 1102）。该请求包含被隔离环境 260 认为是虚拟键值名称的键值名称。使用可应用的规则、使用完整的虚拟化来尝试打开所请求的键值，即，如上所述使用合适的用户和安装范围（步骤 1104）。如果访问被拒绝（步骤

1106)，则将访问拒绝错误返回给请求者（步骤 1109)。如果访问被许可（步骤 1106)，并且所请求的键值被成功打开（步骤 1110)，则将所请求的键值返回给请求者（步骤 1112)。然而，如果访问被许可（步骤 1106)，而所请求的键值没有成功打开（步骤 1110)，则如果所请求键值的父键值也不存在（步骤 1114)，则将适合于请求语义的错误发布给请求者（步骤 1116)。另一方面，如果在使用合适的用户和安装范围的整个虚拟化视图中找到所请求键值的父键值（步骤 1114)，该规则随后确定如何处理该键值操作（步骤 1118)。如果规则动作是“重定向”或者“忽略”（步骤 1120)，则根据该规则将虚拟键值名称直接映射成文本键值名称。具体地，如果规则动作是“忽略”，将文本键值名称被识别为正是虚拟键值名称。如果规则动作是“重定向”，则按照该规则所指定的那样根据该虚拟键值名称确定文本键值名称。随后将建立该文本键值的请求传递给操作系统 210，并将结果返回给请求者（步骤 1124)。另一个方面，如果步骤 1120 中确定的规则动作是“隔离”，则文本键值名称被识别为用户范围中该虚拟键值名称的实例。如果文本键值已经存在，但是其和用于指示其是占位符或者其已经删除的元数据相关联，则修改所关联的元数据以移除那些指示，并且确保该键值是空的。在这两种情况任一中，打开该文本键值的请求被传递给操作系统 210（步骤 1126)。如果文本键值被成功打开（步骤 1128)，则将该文本键值返回给请求者（步骤 1130)。另一方面，如果在步骤 1128 中所请求的键值不能打开，则为当前在用户范围中并不存在的该文本键值的每个祖先建立占位符（步骤 1132)，并且将使用文本名称建立文本键值的请求传递给操作系统 210 并且将结果返回给请求者（步骤 1134)。

[0305] 继续参考图 17 并且更详细地，接收或者拦截用来建立键值的请求（步骤 1102)。在一些实施例中，由替代用于建立键值的一个或多个操作系统 210 函数的函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以被加载到建立进程时该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派键值操作的请求中使用的操作系统 210 资源相关联。对于为每一类键值操作提供单独操作系统 210 函数的实施例，每个函数可以被单独地钩挂。或者，单个钩子函数可以提供来拦截对于多种键值操作的建立或者打开调用。

[0306] 该请求包含被隔离环境 260 处理认为是虚拟键值名称的键值名称。在一些实施例中，将虚拟键值名称被表达为父键值的句柄和到子孙键值的相对路径名称的组合。该父键值句柄和其自身与虚拟键值名称相关联的文本键值名称相关联。请求者可以使用可应用的规则、利用完整的虚拟化来尝试打开该虚拟键值，即，如上所述使用合适的用户和安装范围（步骤 1104)。如果在完整的虚拟化打开操作期间访问被拒绝（步骤 1106)，则将访问拒绝错误返回给请求者（步骤 1109)。如果访问被许可（步骤 1106)，并且所请求的虚拟键值被成功打开（步骤 1110)，则将对应的文本键值返回给请求者（步骤 1112)。然而，如果访问被许可（步骤 1106)，而虚拟键值没有成功打开（步骤 1110)，则已经确定该虚拟键值不存在。如上所确定的，如果所请求的虚拟键值的虚拟父键值也不存在（步骤 1114)，则将适合于请求语义的错误发布给请求者（步骤 1116)。另一方面，如果在使用合适的用户和安装范围的整个虚拟视图中找到所请求的虚拟键值的虚拟父键值（步骤 1114)，则随后通过查阅规则引擎定位一个用于确定如何处理该建立操作的规则（步骤 1118)。在一些实施例中，规则引擎可以被提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈

希表或者平面键值数据库。在一些实施例中，为所请求的键值提供的虚拟键值名称在规则引擎中被用于定位可以应用到该请求的规则。在这些实施例的特定实例中，对于一个特定键值可以在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟键值名称匹配的规则是应用到该请求的规则。在一些实施例中，使用进程标识符来在规则引擎中定位应用到该请求的规则（如果其存在的话）。和请求相关联的规则可以忽略该请求，重定向该请求，或者隔离该请求。尽管图 17 所示为单个数据库事务处理或者对键值的单次查找，但是规则查找也可以实现为一系列规则查找。

[0307] 如果规则动作是“重定向”或者“忽略”（步骤 1120），则根据该规则将虚拟键值名称直接映射成文本键值名称（步骤 1124）。如果规则动作是“重定向”（步骤 1120），则按照规则所指定的那样根据该虚拟键值名称确定文本键值名称（步骤 1124）。如果规则动作是“忽略”（步骤 1120），则将文本键值名称确定为正是虚拟键值名称（步骤 1124）。如果规则动作是“忽略”或者规则动作是“重定向”，则将使用所确定文本键值名称建立文本键值的请求传递给操作系统 210，并将来自操作系统 210 的结果返回给请求者（步骤 1124）。例如，建立名称为“key_1”的虚拟键值的请求可以导致建立名称为“Different_key_1”的文本键值。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的（步骤 1124）。在其他实施例中，操作系统 210 可以提供类似于文件系统过滤器驱动工具的注册表过滤器驱动工具。在这些实施例中，通过发送信号令注册表过滤器管理器使用所确定的文本键值名称解析该请求来对该建立虚拟键值的原始请求做出响应，从而实现对立文本注册表键值的建立。

[0308] 如果步骤 1120 中所确定的规则动作不是“忽略”或者“重定向”而是“隔离”，则文本键值名称被识别为用户范围中的该虚拟键值名称的实例。如果文本键值已经存在，但是其和用于指示其是占位符或者其已经被删除的元数据相关联，则修改所关联的元数据以移除那些指示，并且确保该键值是空的。

[0309] 在一些实施例中，关于注册表键值的元数据可以保存在由该键值保持的特别的值中，该值的存在对于对注册表 API 的一般应用程序使用而言是隐藏的。在一些实施例中，关于注册表键值的少量元数据可以直接保存在文本键值名称中，诸如通过在虚拟名称之后添加元数据指示符，其中元数据指示符是和特定元数据状态相关的唯一串。元数据指示符可以指示或者编码一个或者多个元数据位。由于元数据指示符的存在，以虚拟键值名称访问键值的请求检查文本键值名的各个可能的变化，并且用于获取该键值自身名称的请求被钩挂或者拦截以便使用该文本名称做出响应。在其他实施例中，元数据指示符可以被编码在子键值名称或者注册表值名称中，而不是键值名称自身中。仍在其他实施例中，注册表键值系统可以直接为每一键值提供保存一些第三方元数据的能力。在一些实施例中，元数据保存在数据库或者与注册表数据库分离的其它知识库中。在一些实施例中，单独的子范围可以用来保存被标以删除的键值。该子范围中键值的存在指示该键值被标以删除。

[0310] 在特定一些这样的实施例中，可以维持和查找删除的键值或者键值系统元件的列表，以优化对于所删除的键值的检查。在这些实施例中，如果重新建立删除的键值，则该键值名称可以从删除键值的列表中移除。在其他的这样的实施例中，在该列表变得超过特定尺寸时，将键值名称从该列表中移除。

[0311] 在这两种情况任一中，将打开用户范围文本键值的请求传递给操作系统 210（步

骤 1126)。在一些实施例中，规则可以指定对应于虚拟键值的文本键值应该在除了用户范围之外的范围中建立，诸如安装范围、系统范围、用户子范围或者安装子范围。

[0312] 如果文本键值被成功打开（步骤 1128)，则文本键值返回给请求者（步骤 1130)。如果另一方面，在步骤 1128 中，所请求的键值不能打开，则为当前在用户范围内不存在的文本键值的每一个祖先建立占位符（步骤 1132)，并且将使用文本名称建立文本键值的请求传递给操作系统 210 并且将结果返回给请求者（步骤 1134)。

[0313] 该实施例用于具有每次调用 / 激活仅支持一级建立的 API 或者工具的操作系统 210。而延伸到每次调用 / 激活进行多级建立对于本领域内的普通技术人员来说是显而易见的。

[0314] 图 18 中示出在图 14 所示过程中调用的子例程 2701 的一个实施例。该子例程是在图 14 中所述方法 801 中执行的方法，用来在包括在隔离环境 260 中的一个或者多个安装环境上循环执行。当图 14 中的方法 801 的步骤 816 中确定用户范围候选键值的存在具有中性存在时，调用子例程 2701 并且在第一安装范围中识别候选键值（步骤 2724)。随后确定在第一安装范围中是否找到候选键值（步骤 2702)。如果找到候选键值，则检查该找到的候选键值是否具有否定存在，该否定存在指示该找到的键值已经被标记成删除或者以其他方式不再出现在安装范围中（步骤 2704)。当候选键值名称具有否定存在，进行检查以确定在隔离环境 260 中是否存在更多的安装范围（步骤 2710)，然而，如果候选键值不具有否定存在，则进行检查以确定该候选键值是否具有中性存在（步骤 2706)。当候选键值具有中性存在，该中性存在指示该键值可以是占位符或者该键值可以不存在，并且进一步分析隔离范围 260 以确定隔离环境 260 中是否存在附加的安装范围（步骤 2710)。如果候选键值名称不具有中性存在，则将指示候选键值具有肯定存在的数值返回给主方法 801（步骤 2708)。如果在步骤 2724 之后，确定候选键值没有被找到（步骤 2702)，则分析隔离环境 260 以确定在隔离环境 260 中是否存在附加的安装范围（步骤 2710)。当确定存在附加的安装范围时，在下一个安装范围中识别第二候选键值名称（步骤 2712)。一旦分析下一个安装范围，则确定在该安装范围中是否找到候选键值（步骤 2714)。当找到候选键值时，随后检查该键值以确定其是否具有否定存在（步骤 2704)，或者中性存在（步骤 2706)。如果没有找到第二候选键值，则检查是否存在附加的安装范围（步骤 2710)。当识别出没有更多的附加安装范围时（步骤 2710) 时，检查在最后一次检查期间是否找到候选键值名称（步骤 2724，步骤 2712)，并且如果没有找到键值名称，则将一个否定值返回给主方法 811（步骤 2722)。如果找到一个候选键值名称（步骤 2716)，则确定所找到的候选键值名称是否具有中性存在（步骤 2718)。当候选键值名称具有中性存在时，将中性值返回给主方法 811（步骤 2720)，并且当候选键值名称不具有中性存在时，将否定值返回给主方法 811（步骤 2722)。

[0315] 继续参考图 18 并且更详细地，在子例程 2701 的一个实施例中，在第一安装范围中识别出一个候选键值名称（步骤 2724)。其他实施例可以包括在最后一个安装范围或者安装范围的位置是用户限定值或者预定值的另一个安装范围中检查候选键值的子例程 2701。另一些其他实施例包括子例程 2701，其中，第一安装范围是隔离环境 260 中最高优先级安装范围的安装范围。子例程 2701 的实施例可以包括 ：作为隔离环境 260 中最低优先级安装范围的第一安装范围，在安装范围的子范围中具有最高优先级的第一安装范围，或者在安装范围的子范围中具有最低优先级的第一安装范围。随后确定在第一安装范围中是否找到

候选键值名称（步骤 2702）。在过程 2701 的一个实施例中，当在第一安装范围中找到候选键值，则确定该键值名称是具有否定存在（步骤 2704）、中性存在（步骤 2706）还是肯定存在（步骤 2708）。当键值名称具有否定存在，则进一步确定是否存在附加的安装范围（步骤 2710），如果不存在其他安装范围，则验证找到了候选键值（步骤 2716），并且确定候选键值名称是中性（步骤 2718）还是否定（步骤 2722）。当键值名称不具有否定存在，确定该键值名称是否具有中性存在（步骤 2706），并且如果候选键值不具有中性存在（步骤 2706），则将指示候选键值具有肯定存在的肯定值返回给主方法 811（步骤 2708）。如果键值名称具有中性存在，则检查附加的安装范围是否包括在隔离环境 260 中（步骤 2710），如果不存在其他安装范围，则验证找到了候选键值名称（步骤 2716）并且确定该候选键值名称是否是中性（步骤 2718）或者否定的（步骤 2722）。

[0316] 在一个实施例中，当在第一安装范围中没有找到候选键值时，确定在隔离环境 260 中是否存在附加的安装范围（步骤 2710）。其他实施例可以被配置成搜索预定数量的安装范围。当存在附加的安装范围时，搜索下一个安装范围以识别候选键值名称（步骤 2712）。在一个实施例中，下一个安装范围是优先级低于之前检查的安装范围的优先级的安装范围。例如，在此实施例中，如果之前检查的安装范围是具有最高优先级的第一安装范围，则下一个安装范围可以是具有次高优先级的安装范围。当不存在附加的安装范围时，确定之前是否识别出一个候选键值名称（步骤 2716）。在一个实施例中，之前识别出的候选键值包括在检查附加的安装范围的存在之前由方法 811 识别出的键值。

[0317] 当在隔离环境 260 中包括附加的安装范围时，该方法试图识别出每一安装范围中的候选键值 (2712)。在一个实施例中，确定是否找到候选键值（步骤 2714），并且如果找到了候选键值，则分析该文件以确定该键值是具有定存在（步骤 2704）、中性存在（步骤 2706）还是肯定存在（步骤 2708）。当没有找到候选键值（步骤 2714），则对更多的安装范围进行检查（步骤 2710），并且如果不存在附加的安装范围，则验证找到了候选键值（步骤 2716）。如果没找到候选键值，则返回否定值（步骤 2722），并且如果找到了候选键值，则确定该候选键值是否是中性的（步骤 2718）。

[0318] 图 19 中所示为图 16 中的方法 1003 所调用的子例程 2741 的一个实施例。在一个实施例中，通过图 16 中的过程 1003 调用子例程 2741，以在包括在隔离环境 260 中的所有安装范围上循环执行，并且识别出具有肯定存在的安装范围。在将所列举的系统范围结果保存在工作数据存储之后（步骤 1014），识别出第一安装范围的候选者（步骤 2742）。确定第一安装范围的候选者是否具有否定存在（步骤 2744）。当第一安装范围具有肯定存在时，将肯定值返回给主过程（步骤 2746），并且安装范围被列举到工作数据存储中（步骤 1016）。当第一安装范围具有否定存在时，检查隔离环境 260 中附加安装范围的存在（步骤 2748）。如果不存在附加的安装范围，则将否定值返回给主过程 1003（步骤 2754）。如果存在附加的安装范围，则识别下一个候选安装范围（步骤 2750）并且确定下一安装范围是否具有否定存在（步骤 2744）。验证下一个候选安装范围。

[0319] 继续参考图 19 并且更详细地，当从主过程 1003 调用子例程 2741 时，在图 16 中所示的实施例中，识别第一安装范围的候选者（步骤 2742）。在一个实施例中，第一安装范围的候选者是与包括在隔离环境 260 中的所有其他安装范围相比具有更高优先级的安装范围。其他实施例包括的第一安装范围可以是与包括在隔离环境 260 中的所有其他安装范围

相比具有更低优先级的安装范围。另一些其他实施例可以包括具有比其他安装子范围更高优先级的第一安装子范围，或者具有比包括在子隔离环境 260 中的其他安装范围更高优先级的第一安装范围。

[0320] 过程 2741 的实施例确定安装范围是否具有否定存在（步骤 2744）。在一个实施例中，当第一安装范围确实具有否定存在时（步骤 2744），检查隔离环境 260 是否包括附加的安装范围（步骤 2748）。在另一个实施例中，当安装范围不具有否定存在时（步骤 2744），将肯定值返回给主过程 1003（步骤 2746）。

[0321] 方法 2741 的一个实施例确定隔离环境 260 中是否存在附加的安装范围（步骤 2748）。在一个实施例中，当不存在附加的安装范围时，则将否定值返回给主过程 1003（步骤 2746），而在其它实施例中，当在隔离环境 260 中存在附加的安装范围时，识别下一个候选安装范围（步骤 2750）。

[0322] 命名对象虚拟化

[0323] 可以使用上述技术虚拟化的另一类系统范围资源是命名对象，其包括信号量、互斥体、突变体 (mutant)、等待计时器、事件、工作对象、区段、命名管道、邮槽 (mailslot) 或者可以是用于应用程序的资源的任一其他对象。这些对象的特征在于典型地仅存在于创建它们的进程期间。这些对象的命名空间可在整个计算机（全局范围）或者仅在单个用户会话（会话范围）中有效。

[0324] 现在参考图 20，总的来说，接收或者拦截用来建立或打开命名对象的请求（步骤 1202）。该请求包含隔离环境 260 认为是虚拟名称的对象名称。确定用于确定如何处理该请求的规则（步骤 1204）。如果规则指定该请求应该被“忽略”（步骤 1206），则文本对象名称确定为虚拟名称（步骤 1207），并且将建立或者打开文本对象的请求发布给操作系统 210（步骤 1214）。如果所确定的规则不是忽略该请求，而是指示该请求应该被“重定向”（步骤 1208），则按照重定向规则指定的那样根据该虚拟名称确定文本对象名称（步骤 1210）并且将建立或者打开文本对象的请求传递给操作系统（步骤 1214）。如果规则没有指示该请求不应该重定向（步骤 1208），而是指示该请求应该“隔离”，则按照该隔离规则所指定的那样根据该虚拟名称确定文本对象名称（步骤 1212）并且将建立或者打开文本对象的命令传递给操作系统 210（步骤 1214）。操作系统 210 响应于所发布的建立或者打开命令而返回的文本对象的句柄被返回到请求建立或者打开该虚拟对象的程序（步骤 1216）。

[0325] 继续参考图 20 并且更详细地，拦截来自进程的用来建立或打开命名对象的请求（步骤 1202）。该命名对象可以属于会话范围，其也可以属于全局范围。在一些实施例中，由替代用于创建或者打开命名对象的一个或多个操作系统 210 函数的函数来钩挂该请求。在另一个实施例中，使用钩子动态链接库来拦截该请求。钩子函数可以在用户模式或者内核模式中执行。对于钩子函数在用户模式中执行的实施例，钩子函数可以加载到在建立进程时该进程的地址空间。对于钩子函数在内核模式中执行的实施例，钩子函数可以和在分派系统对象的请求中使用的操作系统 210 资源相关联。建立或者打开命名对象的请求可以涉及用于进程间通信和同步的各种各样的系统范围资源中的任一个，这些系统资源被标以唯一的标识符，其包括信号量、互斥体、互斥门 (mutant)、等待计时器、文件映射对象、事件、工作对象、区段、命名管道、邮槽。对于为每一类对象提供单独的操作系统函数的实施例，每个函数可以被单独地钩挂。或者，可以提供单个钩子函数来拦截对于多种对象的建立或者

打开调用。

[0326] 所拦截的请求包含隔离环境 260 认为是虚拟名称的对象名称。通过查阅规则引擎可确定用于确定如何处理对于该对象的请求的规则（步骤 1204）。在一些实施例中，规则引擎可以提供为关系数据库。在其他实施例中，规则引擎可以是树形结构的数据库，哈希表或者平面文件数据库。在一些实施例中，为所请求的对象提供的虚拟名称被用于在规则引擎中定位可应用到该请求的规则。在这些实施例的特定实例中，对于一个特定的对象可以在规则引擎中存在多个规则，并且在这些实施例中，最长前缀和虚拟名称的相匹配的规则是应用到该请求的规则。在一些实施例中，使用进程标识符来在规则引擎中定位应用程序到该请求的规则（如果其存在的话）。和请求相关联的规则可以忽略该请求，重定向该请求，或者隔离该请求。尽管图 18 所示规则查找为一系列决策，但是规则查找也可以作为单个数据库事务处理发生。

[0327] 如果规则指示该请求应该被“忽略”（步骤 1206），则文本对象名称被确定为就是该虚拟名称，并且将建立或者打开该文本对象的请求传递给操作系统 210（步骤 1214）。例如，建立或者打开名称为“Object_1”的对象的请求可以导致建立名称为“Object_1”的实际对象。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。

[0328] 如果通过访问规则引擎而确定的规则不是忽略该请求，而是相反指示该请求应该被“重定向”（步骤 1208），则按照由重定向规则所指定的那样根据虚拟名称确定文本对象名称（步骤 1210）并且将建立或者打开文本对象的请求传递给操作系统 210（步骤 1214）。例如，建立或者打开名称为“Object_1”的命名对象的请求可以导致建立名称为“Different_object_1”的实际对象。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。

[0329] 如果规则指示该请求不应该重定向（步骤 1208），而相反是指示该请求应该“隔离”，则按照隔离规则所指定的那样根据该虚拟名称确定文本对象名称（步骤 1212）并且将建立或者打开该文本对象的命令发布给操作系统 210（步骤 1214）。例如，建立或者打开名称为“Object_1”的命名对象的请求可以导致建立名称为“Isolated_Object_1”的实际对象。在一个实施例中，这是通过调用被钩挂的函数的原始版本并且将所形成的文本名称作为参量传递到该函数来实现的。

[0330] 用来隔离所请求的系统对象而形成的文本名称可以基于所接收的虚拟名称和范围专用标识符。范围专用标识符可以是和安装范围、用户范围、会话范围或三者的一些组合相关联的标识符。范围专用标识符用来“打乱 (mangle)”在请求中接收到的虚拟名称。例如，如果对于其相关联标识符是“SA1”的安装范围，对于命名对象“Object_1”的请求被隔离，则文本名称可以是“Isolated_AppScope_SA1_Object_1”。下表识别出利用会话范围、用户范围或者安装范围打乱 (mangle) 对象的名称的效果。利用范围组合进行的打乱 (mangle) 结合了表中所列出的限制。

[0331]

	会话专用标识符	用户专用标识符	安装专用标识符

全局对象	对于在用户会话环境中执行的所有隔离的应用程序均可用的对象	对于代表用户执行的所有隔离的应用程序均可用的对象	对于在安装范围中执行的所有隔离的应用程序均可用的对象
会话对象	对于在用户会话环境中执行的所有隔离的应用程序均可用的对象	对于代表用户在会话中执行的所有隔离的应用程序均可用的对象	对于在会话中在安装范围内执行的所有隔离的应用程序均可用的对象

[0332] 对于操作系统 210 是 WINDOWS 家族操作系统的一个的实施例，可以通过切换和对象相关的全局 / 本地名称前缀可修改对象范围，对于隔离的应用程序，这具有和利用会话专用标识符打乱对象名称一样的效果。然而，切换全局 / 本地名称前缀也会影响非隔离应用程序的对象范围。

[0333] 操作系统 210 响应于步骤 1214 中发布的建立或者打开命名对象的命令而返回的文本对象的句柄被返回到请求建立或者打开虚拟对象的程序（步骤 1216）。

[0334] 应用程序专用的用户根目录

[0335] 图 21A 中示出计算机器 200 的一个实施例，该计算机器 200 配置为提供对超过一个的用户 238、240 和 242 的访问，并且具有配置为提供应用程序专用的、用户专用的原生资源的用户范围。所示计算机器 200 包括如下部件：安装在其上的操作系统 210，指示用户动作的用户会话 238、240、242，和在用户会话 238、240、242 中执行的应用程序 202、204、206、206’。操作系统 210 具有系统根目录 216、安装根目录 214 和用户根目录 211。系统根目录 216 中包括的系统范围 256 还包括诸如以下的原生资源 258：文件系统 218、注册表 220 和对象 222。安装根目录 214 包括超过一个的安装范围 244、246、252、248、250、254，其中每一个安装范围连接到另一个范围。每一个安装范围中包括原生资源的实例 258’、258”、258”’、258””、258””’、258”””。用户根目录 212 包括超过一个的用户范围 2103、2106、2109，其中每一个范围和用户会话 238、240、242 相关联，并且每一个范围包括原生资源的实例。操作系统 210 中包括的隔离环境 260 将用户根目录 212 和安装根目录 214 集合在一起。

[0336] 继续参考图 21A 并且更详细地，图 21A 中包括的元件和图 2B 包括的元件基本类似，除了用户根目录 212 包括专用于保存应用程序专用的、用户指定的设置的用户范围 2103、2106、2109。另外一个不同是由第三用户会话中的第三用户执行的应用程序。在此用户会话中，第三用户执行第三应用程序 206、206’的两个实例。在一个实施例中，用户范围 2103、2106、2109 包括对应于应用程序配置文件的原生资源。每个应用程序配置文件包括应用程序专用的设置信息。在计算机器 200 的一个实施例中，在用户会话中执行应用程序的用户可以改变执行的应用程序的设置。所改变的设置被保存在应用程序配置文件中，其还通过包括在用户范围中的一组原生资源来表示。当用户执行该应用程序时，该应用程序根据所改变的应用程序设置来执行。这样的例子包括第一用户在第一用户会话 238 中执行第一应用程序 202。在此例中，对应于第一应用程序的用户范围 2103 保存所有针对第一应用程序的第一用户指定的应用程序设置，并且每当第一应用程序由第一用户执行时使这些设

置对于该第一应用程序是可用的。进一步参考该例子，如果第一用户将在多于一个的隔离环境中执行该第一应用程序，则出现在每一隔离环境中的该第一应用程序的实例将具有相同的应用程序专用的和用户指定的应用程序设置。

[0337] 在计算机器 200 的一个实施例中，包括在第三用户会话 242 中执行第三应用程序的两个实例 206、206’的第三用户。包括专用于第三应用程序的原生资源的用户范围 2109 也包括在此实施例中。在此实施例中，第三应用程序的每一个实例 206、206’被配置为根据用户范围 2109 中包括的那些应用程序专用的和用户指定的资源来在第三用户会话 242 中执行。其他实施例可以包括具有被配置为保存针对多于一个的应用程序的设置信息的子范围的用户范围 2109。

[0338] 图 21A 中描述的计算机器 200 的一个例子包括第一隔离环境中并且和第一应用程序相关联的第一安装范围。第一安装范围中包括配置为执行该第一应用程序的可执行组件 (component)。第二隔离环境中还包括和第一应用程序相关的第二安装范围。第二安装范围中包括配置为执行第一应用程序的可执行部件。用户范围包括在第一隔离环境和第二隔离环境的每一个中，其中用户范围保存具有专用于执行第一应用程序的用户设置信息的应用程序配置文件。选择哪个应用程序专用的用户范围包括在隔离环境中是基于哪个安装范围包括可执行组件。在此例中，第一隔离环境和第二隔离环境的每一个具有包括可执行组件的安装范围，其中该安装范围对应于第一应用程序。基于第一可执行组件的位置，包括在第一隔离环境和第二隔离环境二者中的用户范围是对应于第一应用程序的用户范围。在此例中，以及其它实施例中，选择包括哪个用户范围是指限定该用户范围。基于可执行组件和应用程序之间的关联来限定用户范围，是基于哪个安装配置文件包括可执行组件来确定用户范围对应于哪个应用程序。其他实施例包括基于第一发现的可执行组件来限定用户范围。

[0339] 图 21B 中示出配置为聚集多于一个的隔离范围的虚拟范围 2130 的框图的实施例，用来建立包括在虚拟范围 2130 中的虚拟化原生资源视图。虚拟范围 2130 中是以下隔离范围的聚集 ：对应于第一应用程序的用户应用程序 1 范围 2120，对应于第一应用程序的安装范围 2124A，对应于第二应用程序的安装范围 2124B，对应于第 N 个应用程序的安装范围 2124N，和系统范围 2126。

[0340] 继续参考图 21B 并且更详细地，在虚拟范围 2130 的一个实施例中，包括一个用户范围，其包含第一应用程序的执行所使用的原生资源。在一个实施例中，包含在虚拟范围 2130 中的原生资源是用户指定的应用程序设置，其指导该应用程序根据用户指定的设置来执行。在另一个实施例中，虚拟范围 2130 中包含的原生资源是对应于第一应用程序的应用程序配置文件的应用程序设置。该应用程序配置文件基于用户指定的设置信息来产生并且该配置文件规定第一设置应该执行的方式。在一些实施例中，虚拟范围 2130 包括具有专用于第一应用程序设置信息的子范围和专用于其它应用程序设置信息的附加子范围的用户范围 2120。虚拟范围 2130 的另一些其他实施例包括具有子范围的用户范围，其中该至少一个子范围专用于提供第一应用程序专用的设置信息。

[0341] 图 21B 中所示安装范围 2124A-2124N 基本类似于包括在图 4 中描述的虚拟范围 262 中的那些安装范围 284A-284N。此外，图 21B 中示出的系统范围 2126 基本类似于图 4 中描述的虚拟范围 262 中的系统范围 286。图 4 和图 21B 的差别包括代表应用程序的安装范围 2124A-2124N，使得第一安装范围 2124A 代表第一应用程序，第二安装范围 2124B 代表

第二应用程序，并且第 N 安装范围 2124N 代表第 N 应用程序。

[0342] 图 21C 中示出描述用于聚集在隔离范围中找到的各原生资源的方法 2203 的流程图。该方法 2203 包括拦截第一应用程序产生的请求（步骤 2206），其中该请求用于列举原生资源（步骤 2206）。执行对原生资源的系统范围实例的列举，并且将所有列举的所请求原生资源的实例增加到虚拟原生资源视图中（步骤 2209）。方法 2203 还包括列举包括在每个安装范围中的原生资源实例（步骤 2212），添加那些找到的资源，并且使用新近发现的原生资源实例替换在虚拟原生资源视图中的那些已经存在的原生资源（步骤 2215）。确定是否存在对应于第一应用程序的用户范围（步骤 2218）。如果存在配置为包括第一应用程序专用设置的用户范围，则搜索第一应用程序 / 用户范围以寻找所请求原生资源的实例（步骤 2221）。当没有找到对应于第一应用程序的用户范围时，在现有用户范围中执行搜索以寻找所请求原生资源的实例（步骤 2227）。一旦在非专用的用户范围或者专用于第一应用程序的用户范围中执行搜索，每一个所找到的原生资源实例被增加到虚拟化原生资源视图中并且使用在其中一个用户范围中找到的基本相似的资源替换在虚拟视图中已经存在的那些资源（步骤 2224）。

[0343] 继续参考图 21C 并且更详细地，基本上对应于图 3A 中所示方法 401 的一些方面的方法 2203 中的那些方面可以根据上述方法 401 中的配置或者实施例来实现。方法 2203 中不同于图 3A 中所示方法 401 的方面如下：确定第一应用程序专用的用户范围（步骤 2218），和进一步搜索应用程序专用的用户范围（步骤 2221）。在方法 2203 的一个实施例中，使用过滤器驱动来确定是否存在应用程序专用的用户范围，并且进一步搜索已知的应用程序专用的用户范围。方法 2203 的其他实施例包括的应用程序专用的用户范围包括用户产生的并且保存在第一应用程序专用的用户范围中的第一应用程序设置信息。另一些其他实施例包括具有专用于提供原生资源实例的子范围的用户范围，这些原生资源由用户限定并且由第一应用程序用来根据用户指定的参数执行。其他实施例包括方法 2203，其在第一应用程序执行在相同的用户会话和不同的隔离环境中时搜索第一应用程序专用的范围。

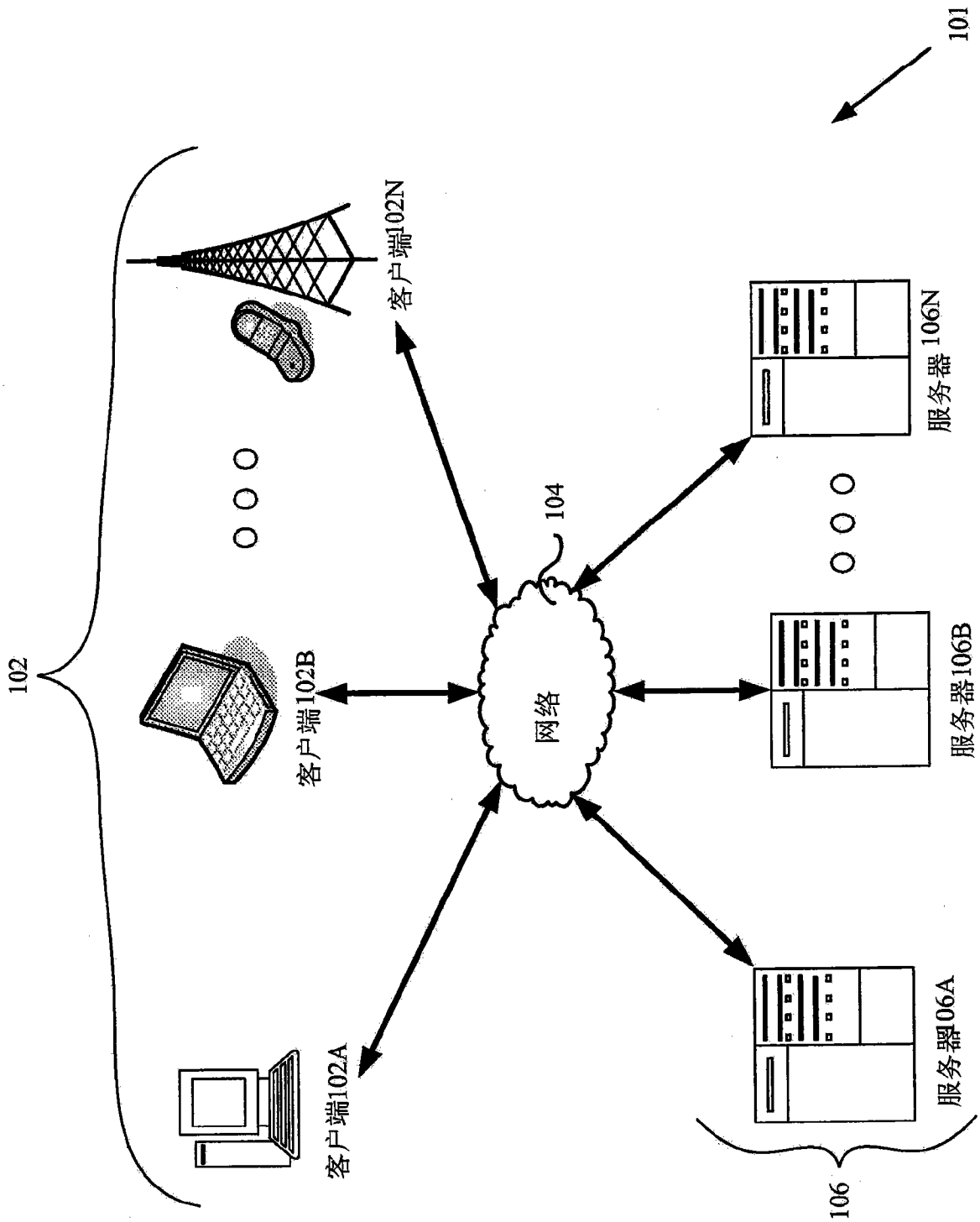

101
102
客户端102N
客户端102B
客户端102A
104
网络
服务器106N
服务器106B
服务器106A
106

图 1A

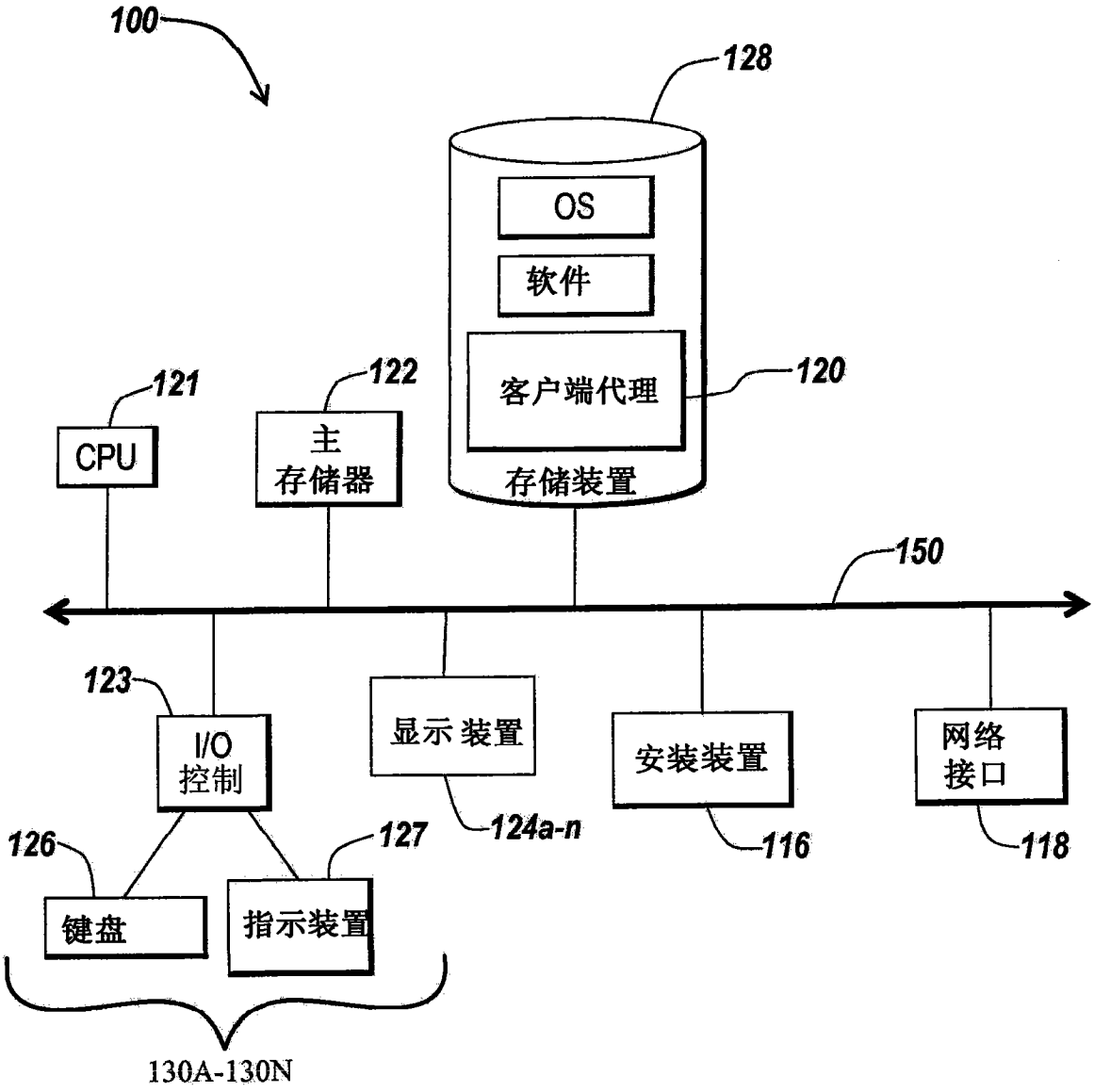

100
128
OS
软件
客户端代理
120
存储装置
121
CPU
122
主
存储器
150
123
I/O
控制
显示 装置
124a-n
安装装置
116
网络
接口
118
126
键盘
127
指示装置
130A-130N

图 1B

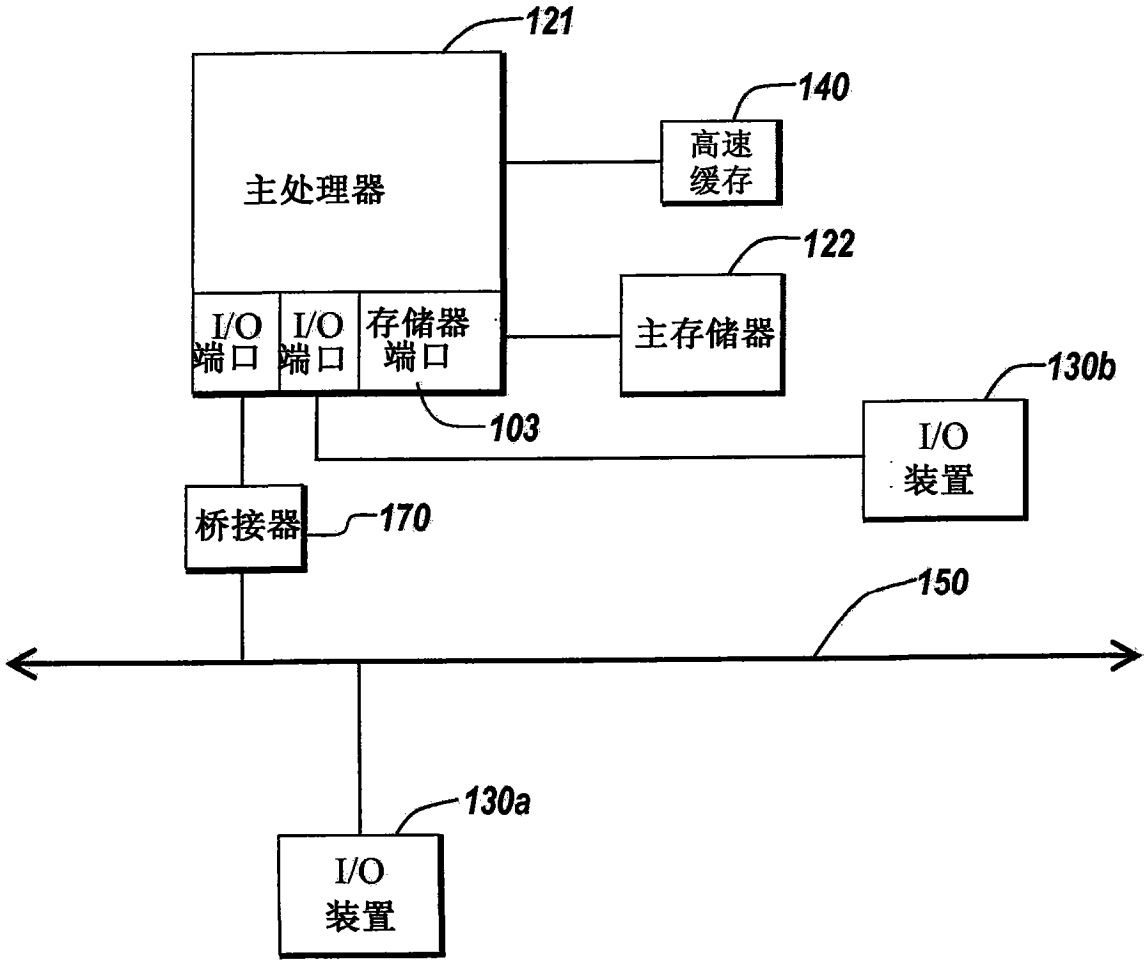

121
主处理器
140
高速
缓存
122
主存储器
I/O
端口
I/O
端口
存储器
端口
103
130b
I/O
装置
桥接器
170
150
130a
I/O
装置

图 1C

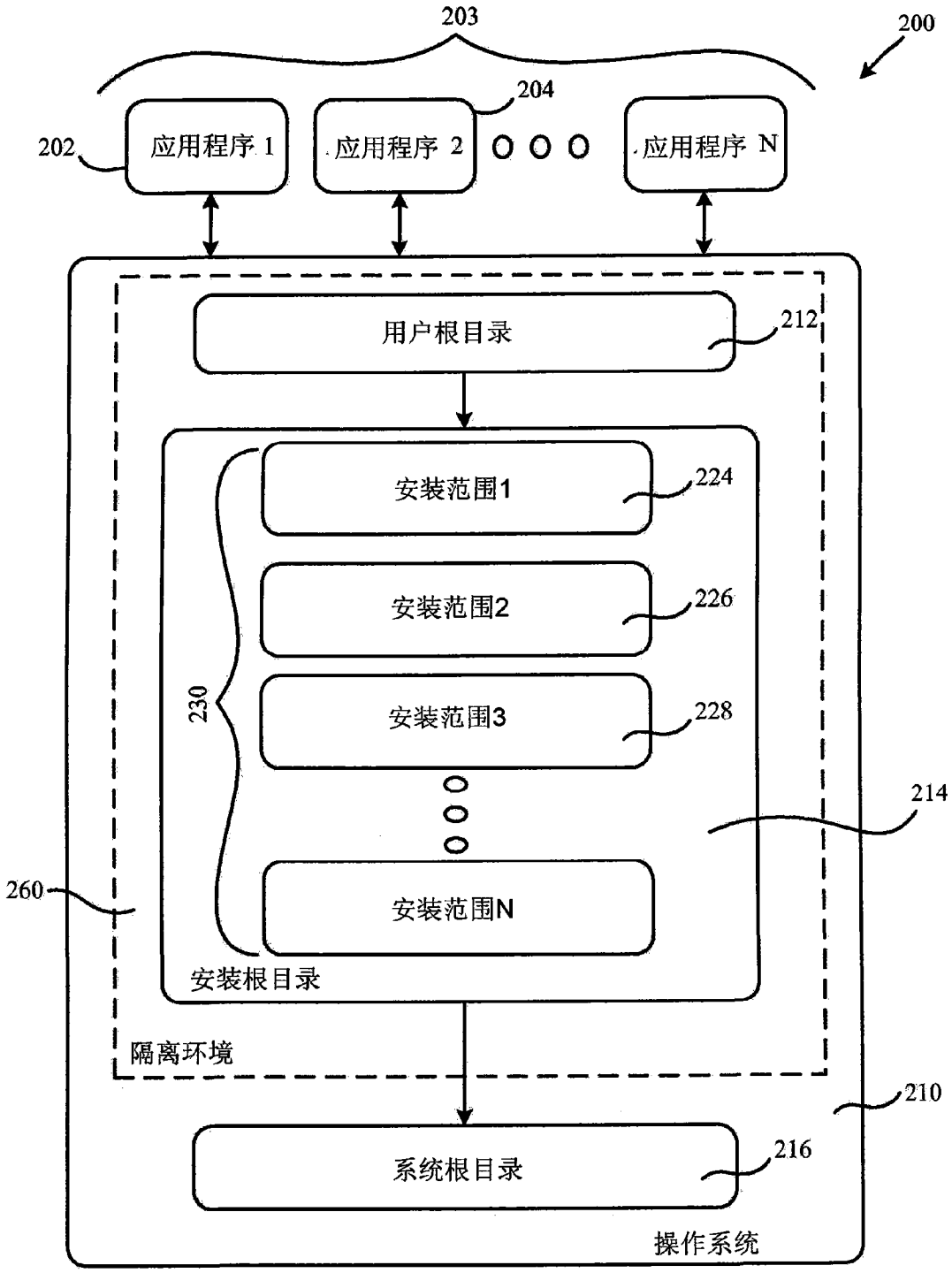

203
200
204
202
应用程序 1
应用程序 2
应用程序 N
用户根目录
212
安装范围1
224
安装范围2
226
安装范围3
228
230
214
260
安装范围N
安装根目录
隔离环境
210
系统根目录
216
操作系统

图 2A

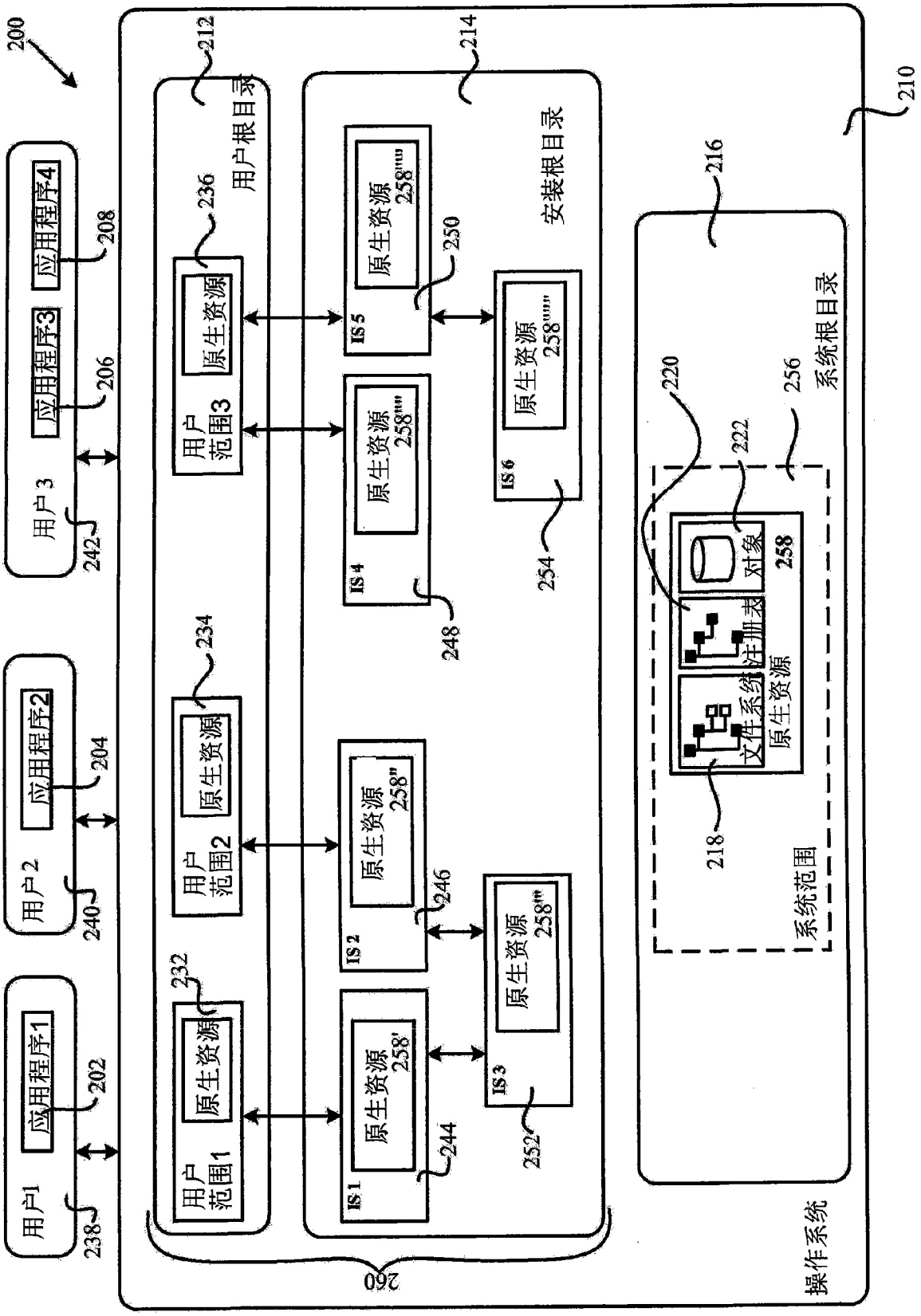

200
用户1
应用程序1
202
238
用户 2
应用程序2
204
240
用户 3
应用程序3
206
应用程序4
208
242
操作系统
210
用户根目录
212
安装根目录
214
系统根目录
216
系统范围
218
220
222
256
文件系统
注册表
原生资源
对象
258
用户范围1
232
原生资源
用户范围2
234
用户范围3
236
原生资源
IS 1
原生资源 258'
244
IS 2
原生资源 258"
246
IS 3
原生资源 258'''
252
IS 4
原生资源 258''''
248
IS 5
原生资源 258'''''
250
IS 6
原生资源 258''''''
254
260

图 2B

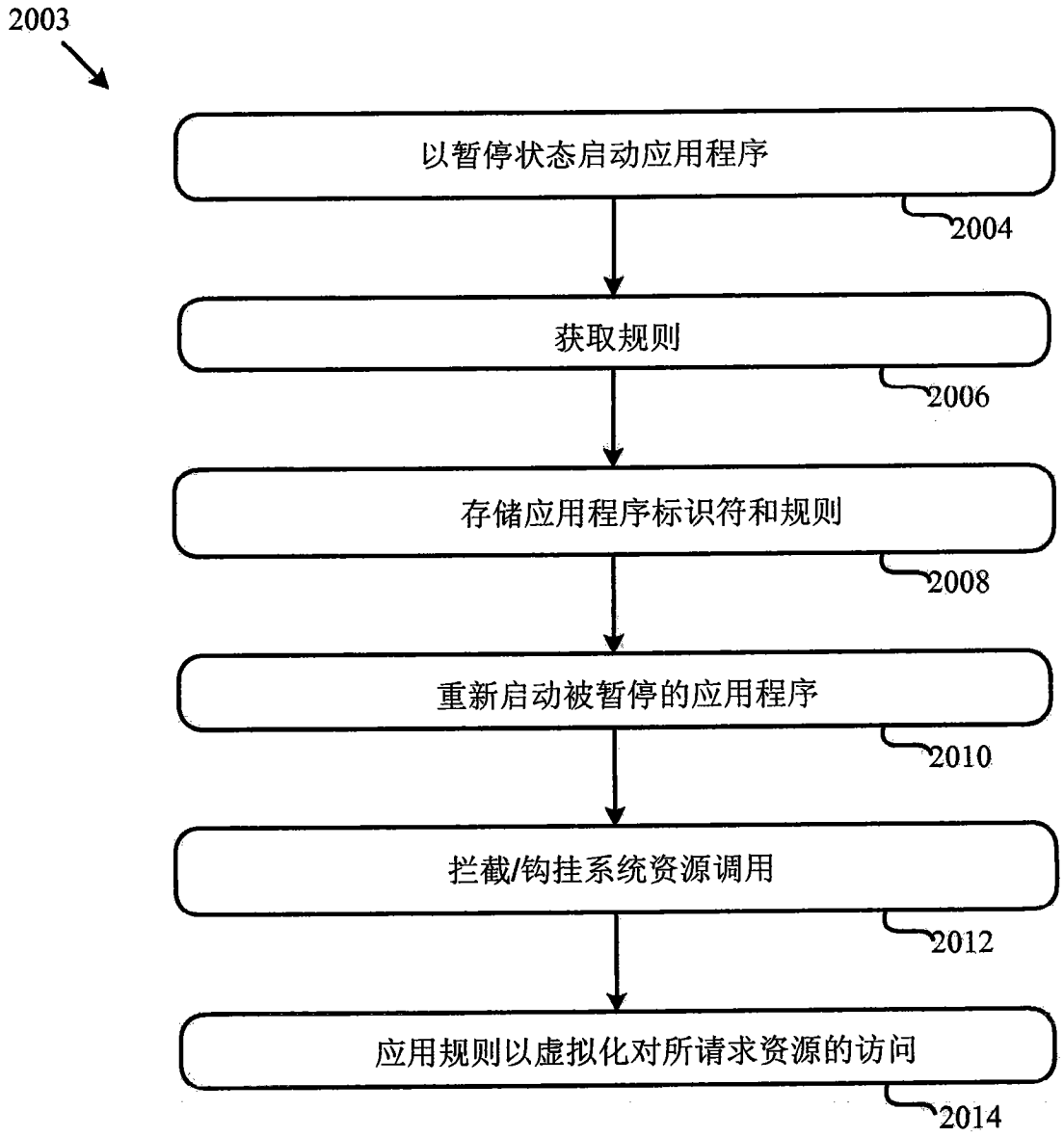

2003
以暂停状态启动应用程序
2004
获取规则
2006
存储应用程序标识符和规则
2008
重新启动被暂停的应用程序
2010
拦截/钩挂系统资源调用
2012
应用规则以虚拟化对所请求资源的访问
2014

图 2C

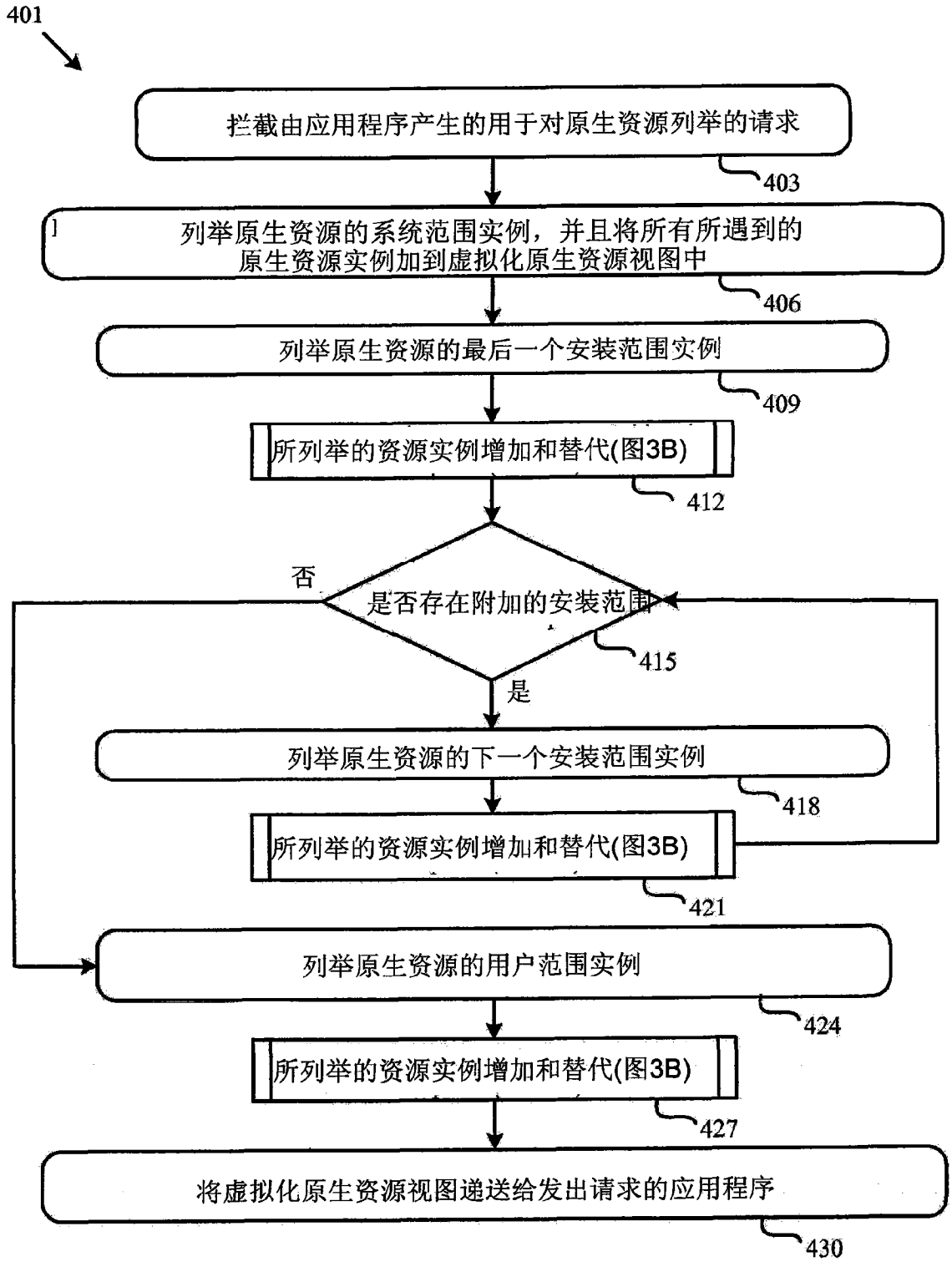

401
拦截由应用程序产生的用于对原生资源列举的请求
403
列举原生资源的系统范围实例，并且将所有所遇到的原生资源实例加到虚拟化原生资源视图中
406
列举原生资源的最后一个安装范围实例
409
所列举的资源实例增加和替代(图3B)
412
否
是否存在附加的安装范围
415
是
列举原生资源的下一个安装范围实例
418
所列举的资源实例增加和替代(图3B)
421
列举原生资源的用户范围实例
424
所列举的资源实例增加和替代(图3B)
427
将虚拟化原生资源视图递送给发出请求的应用程序
430

图 3A

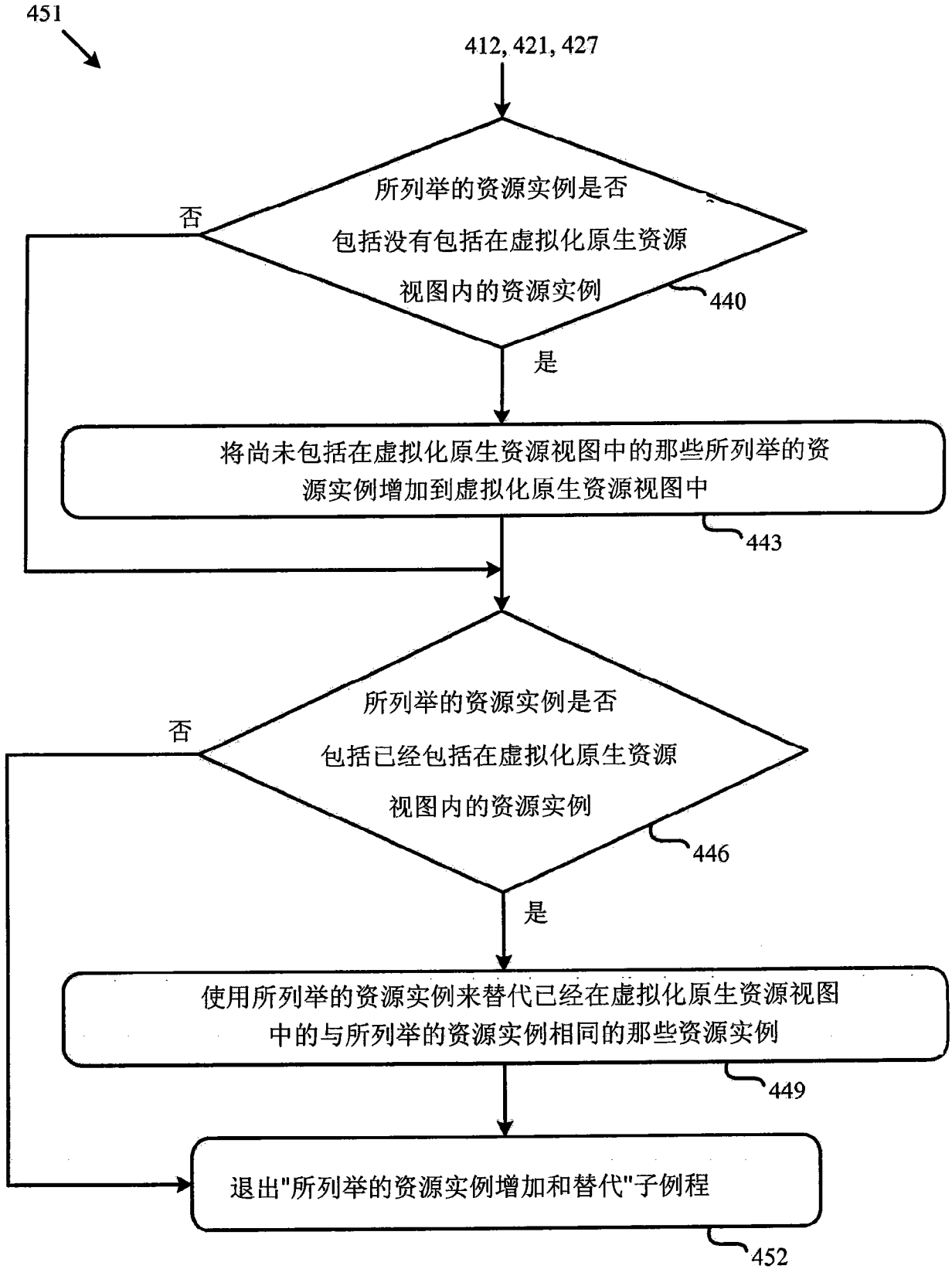

451
412, 421, 427
所列举的资源实例是否包括没有包括在虚拟化原生资源视图内的资源实例
440
否
是
将尚未包括在虚拟化原生资源视图中的那些所列举的资源实例增加到虚拟化原生资源视图中
443
所列举的资源实例是否包括已经包括在虚拟化原生资源视图内的资源实例
446
否
是
使用所列举的资源实例来替代已经在虚拟化原生资源视图中的与所列举的资源实例相同的那些资源实例
449
退出"所列举的资源实例增加和替代"子例程
452

图 3B

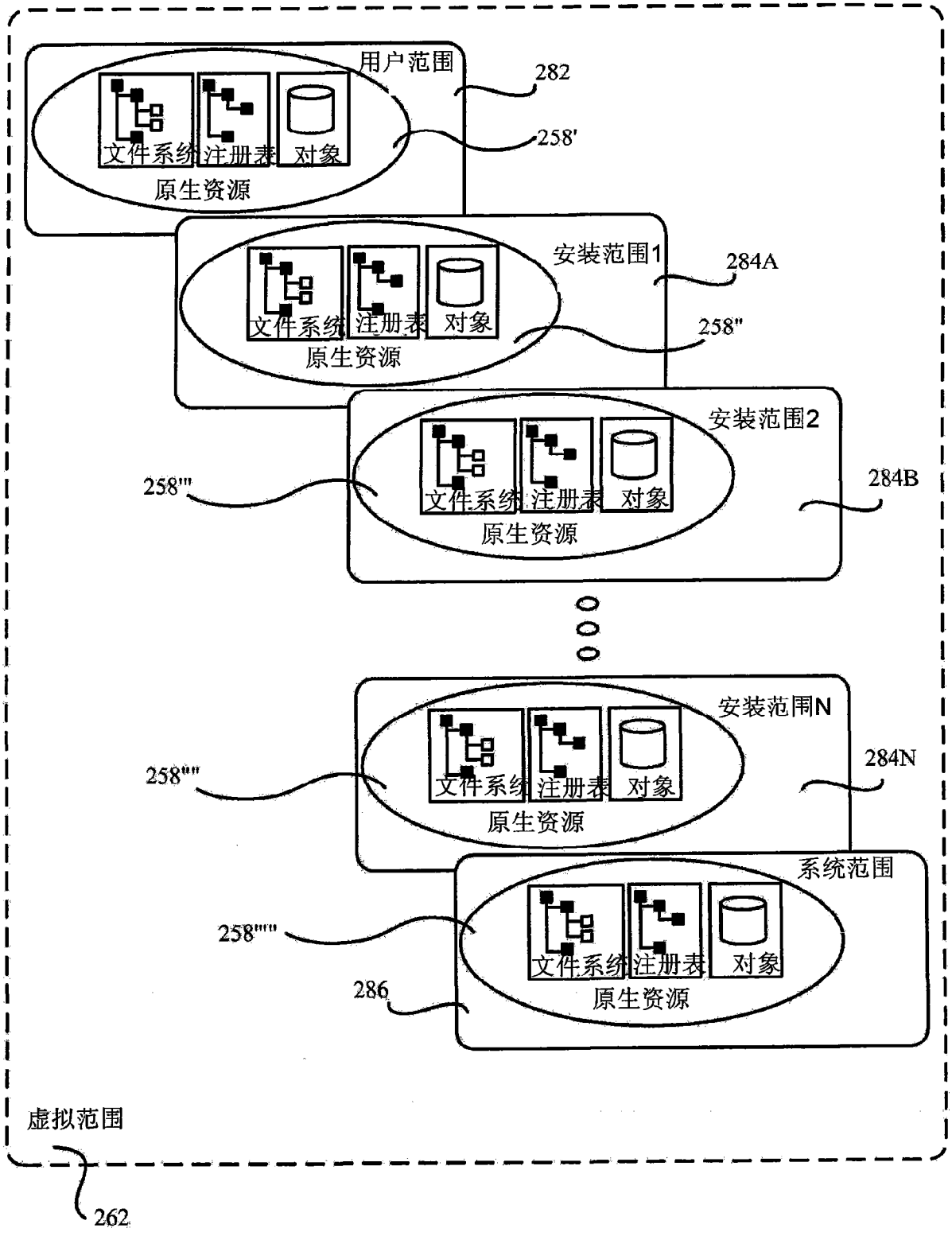

用户范围
282
258'
文件系统
注册表
对象
原生资源
安装范围1
284A
258"
文件系统
注册表
对象
原生资源
安装范围2
258'''
284B
文件系统
注册表
对象
原生资源
安装范围N
258''''
284N
文件系统
注册表
对象
原生资源
系统范围
258'''''
286
文件系统
注册表
对象
原生资源
虚拟范围
262

图 4

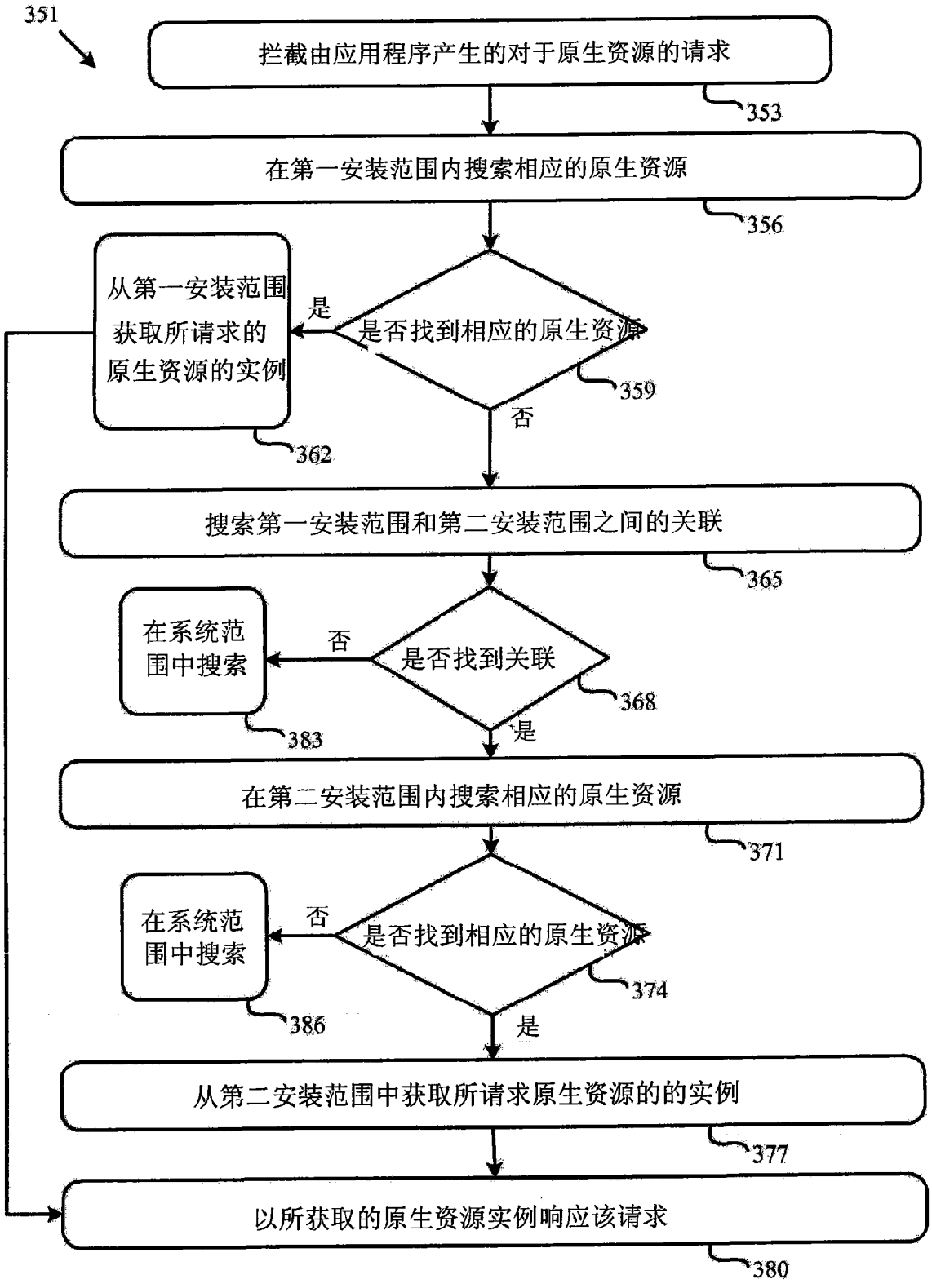

351
拦截由应用程序产生的对于原生资源的请求
353
在第一安装范围内搜索相应的原生资源
356
是否找到相应的原生资源
359
是
从第一安装范围获取所请求的原生资源的实例
362
否
搜索第一安装范围和第二安装范围之间的关联
365
是否找到关联
368
否
在系统范围中搜索
383
是
在第二安装范围内搜索相应的原生资源
371
是否找到相应的原生资源
374
否
在系统范围中搜索
386
是
从第二安装范围中获取所请求原生资源的的实例
377
以所获取的原生资源实例响应该请求
380

图 5

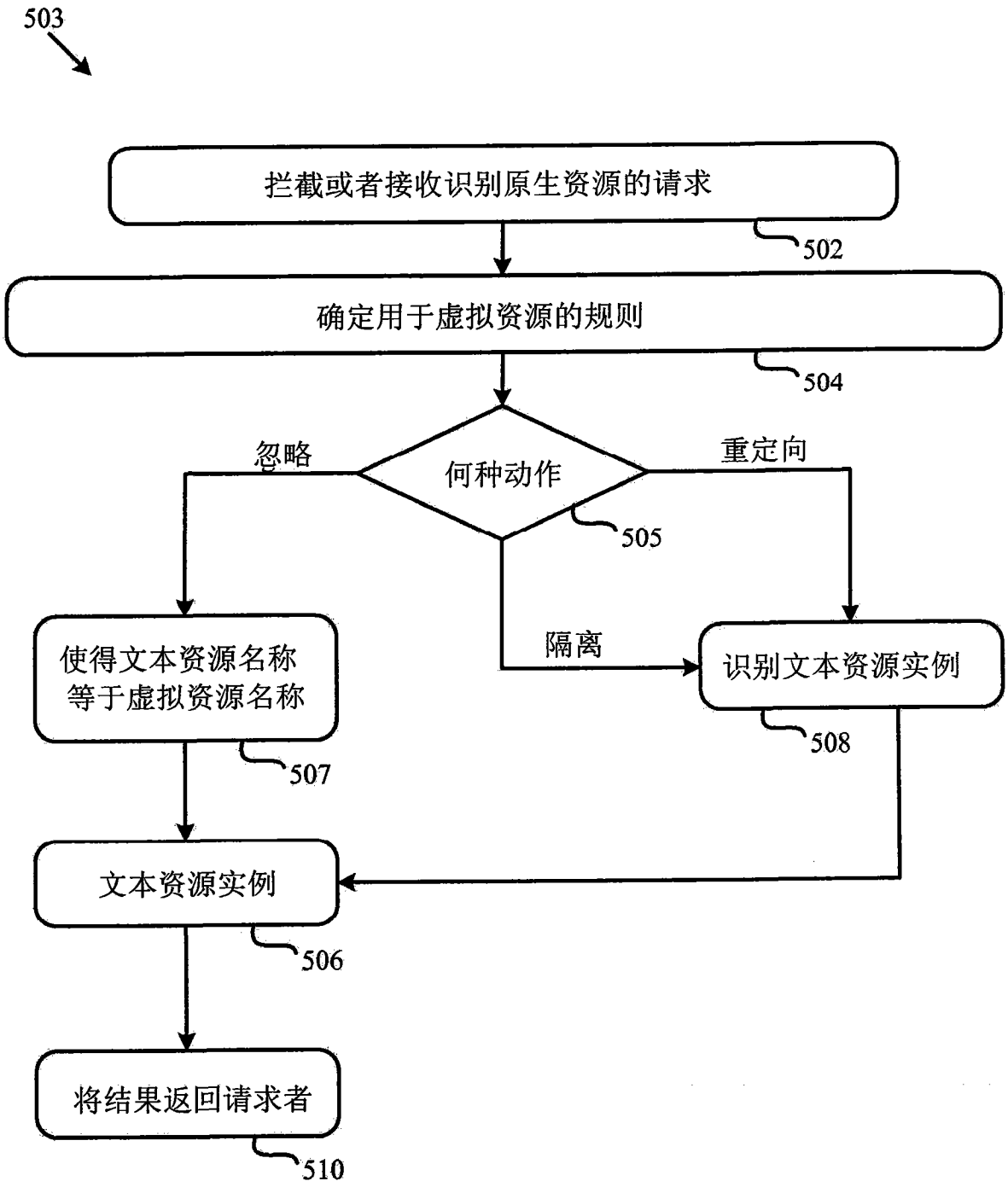

503
拦截或者接收识别原生资源的请求
502
确定用于虚拟资源的规则
504
何种动作
505
忽略
重定向
隔离
使得文本资源名称
等于虚拟资源名称
507
识别文本资源实例
508
文本资源实例
506
将结果返回请求者
510

图 6A

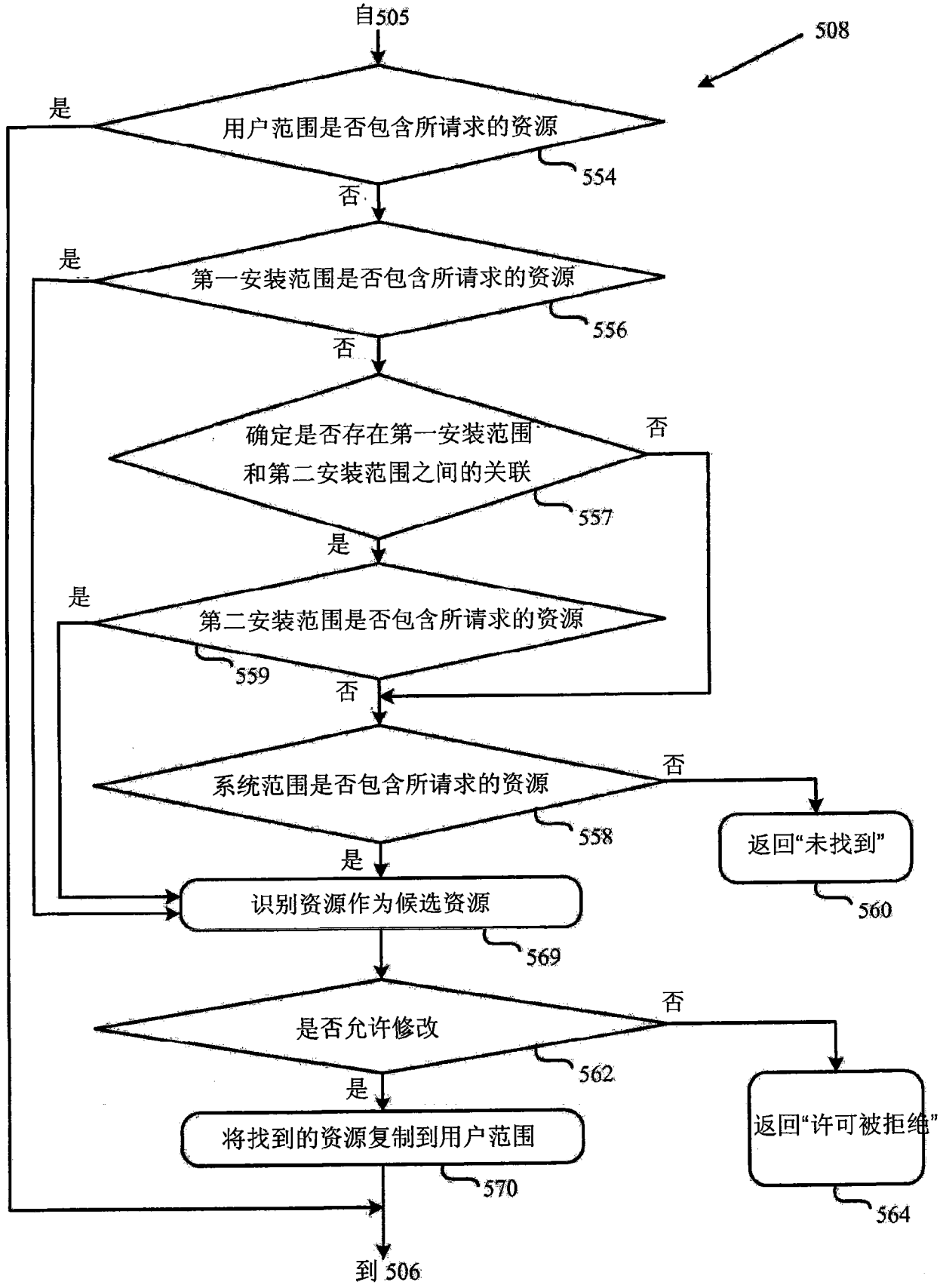

自505
508
用户范围是否包含所请求的资源
554
是
否
第一安装范围是否包含所请求的资源
556
是
否
确定是否存在第一安装范围
和第二安装范围之间的关联
557
否
是
第二安装范围是否包含所请求的资源
559
是
否
系统范围是否包含所请求的资源
558
否
返回“未找到”
560
是
识别资源作为候选资源
569
是否允许修改
562
否
是
将找到的资源复制到用户范围
570
返回“许可被拒绝”
564
到 506

图 6B

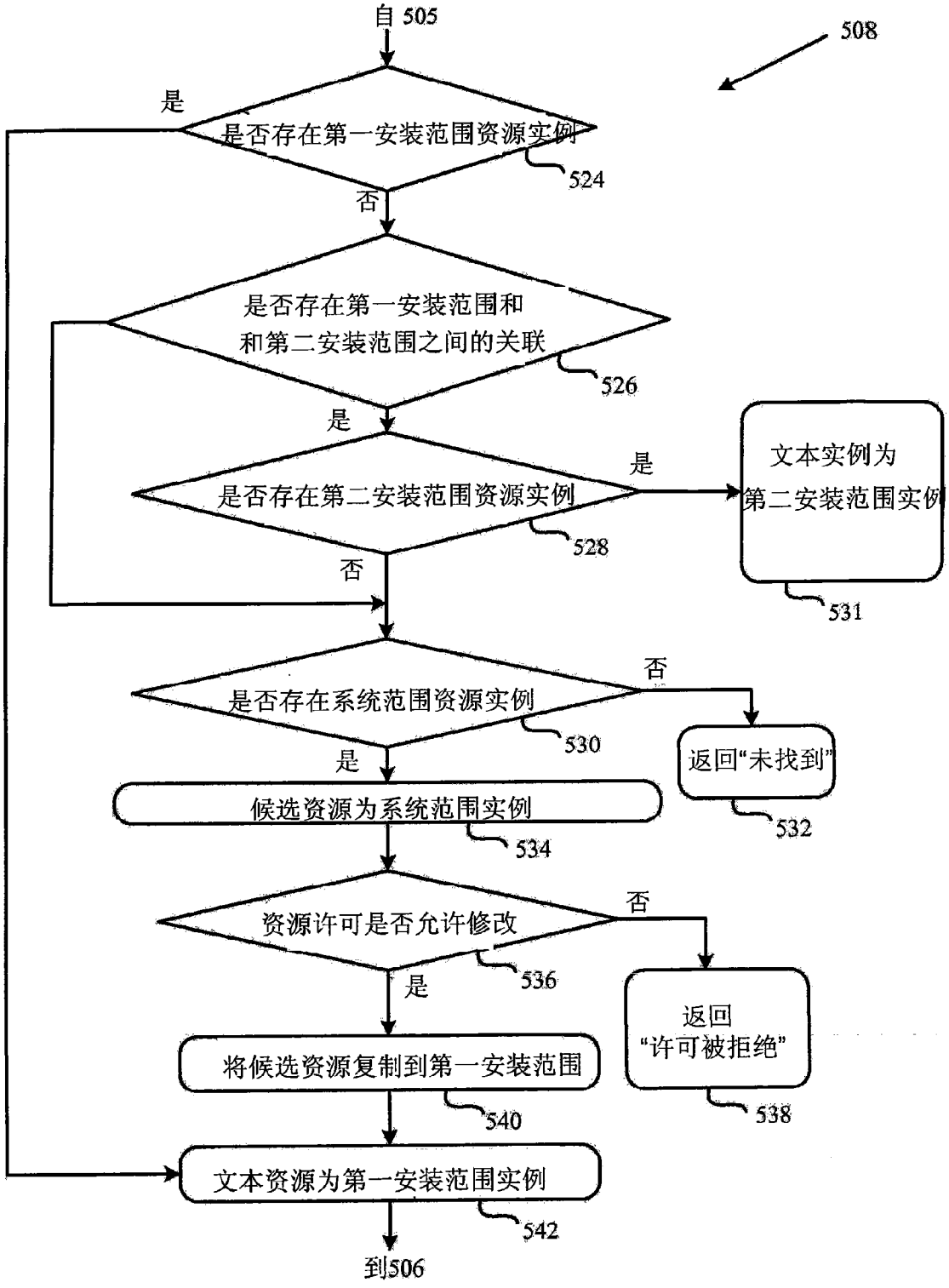

自 505
508
是否存在第一安装范围资源实例
524
是
否
是否存在第一安装范围和
和第二安装范围之间的关联
526
是
是否存在第二安装范围资源实例
528
是
文本实例为
第二安装范围实例
531
否
是否存在系统范围资源实例
530
否
返回“未找到”
532
是
候选资源为系统范围实例
534
资源许可是否允许修改
536
否
返回
“许可被拒绝”
538
是
将候选资源复制到第一安装范围
540
文本资源为第一安装范围实例
542
到506

图 6C

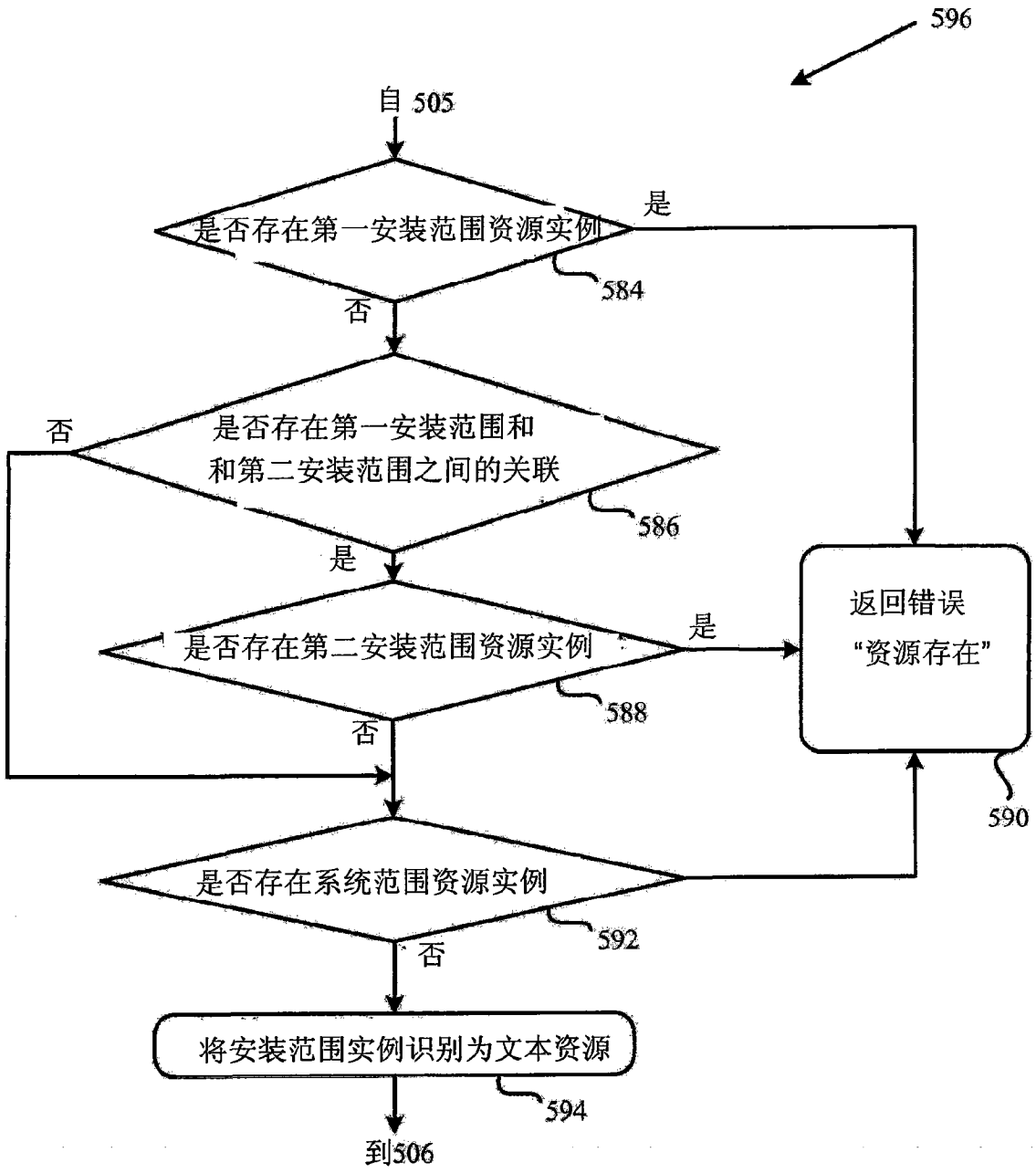

596
自 505
是否存在第一安装范围资源实例
是
584
否
是否存在第一安装范围和
和第二安装范围之间的关联
否
586
是
是否存在第二安装范围资源实例
是
返回错误
“资源存在”
588
否
590
是否存在系统范围资源实例
592
否
将安装范围实例识别为文本资源
594
到506

图 6D

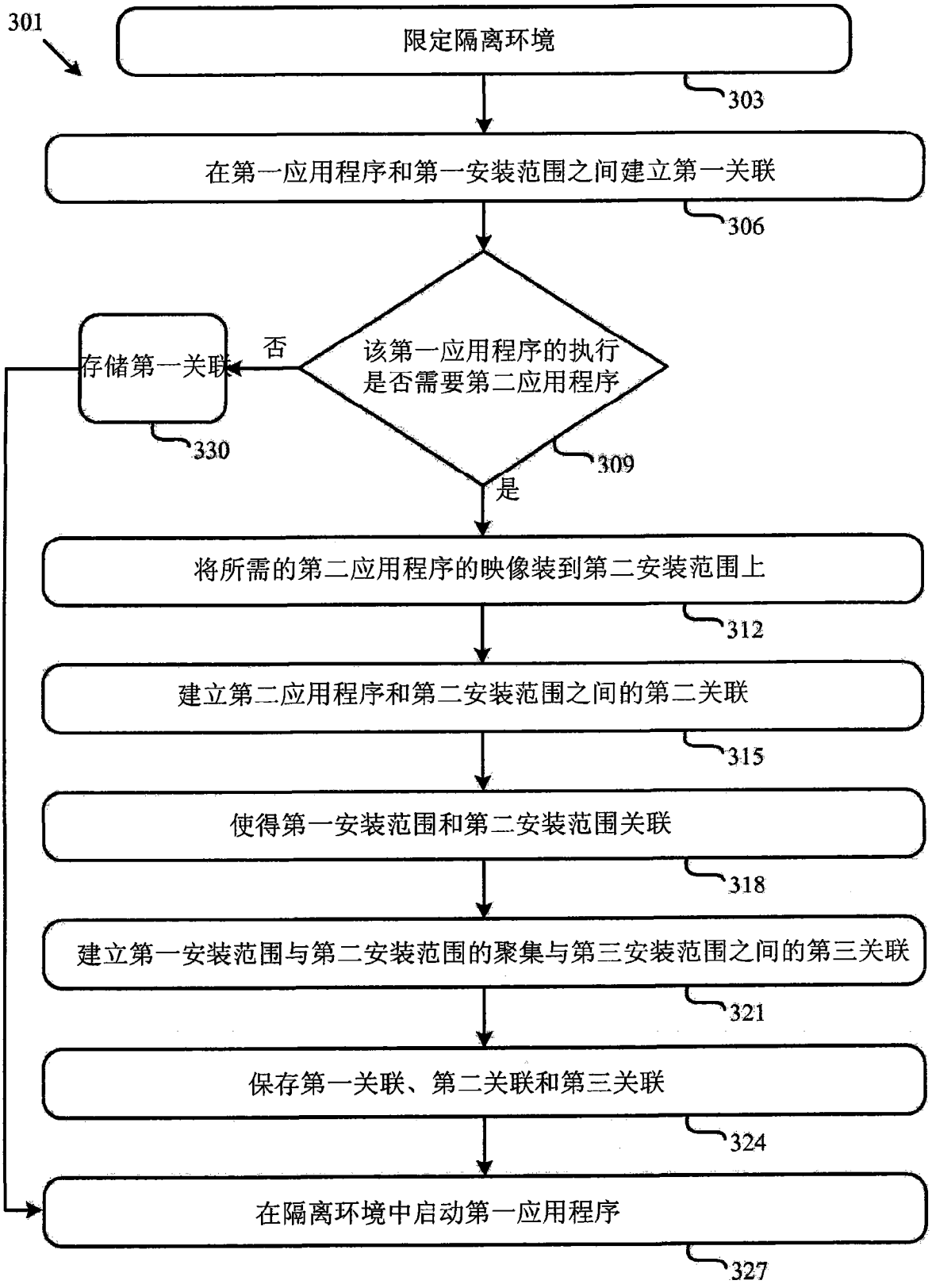

301
限定隔离环境
303
在第一应用程序和第一安装范围之间建立第一关联
306
该第一应用程序的执行
是否需要第二应用程序
309
否
存储第一关联
330
是
将所需的第二应用程序的映像装到第二安装范围上
312
建立第二应用程序和第二安装范围之间的第二关联
315
使得第一安装范围和第二安装范围关联
318
建立第一安装范围与第二安装范围的聚集与第三安装范围之间的第三关联
321
保存第一关联、第二关联和第三关联
324
在隔离环境中启动第一应用程序
327

图 7

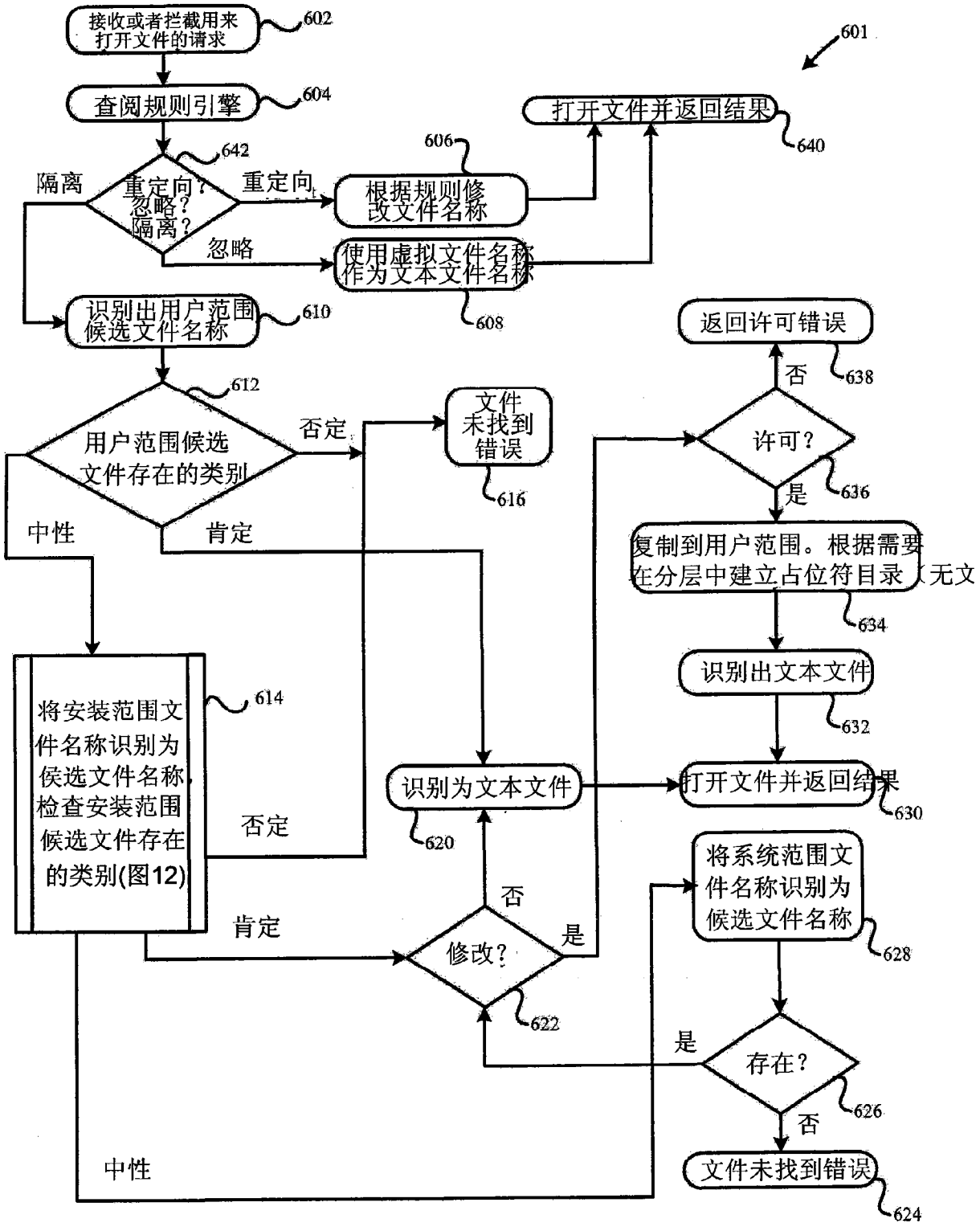

接收或者拦截用来打开文件的请求
602
601
查阅规则引擎
604
打开文件并返回结果
640
642
606
隔离
重定向？忽略？隔离？
重定向
根据规则修改文件名称
忽略
使用虚拟文件名称作为文本文件名称
608
识别出用户范围候选文件名称
610
返回许可错误
638
否
612
用户范围候选文件存在的类别
否定
文件未找到错误
616
许可？
636
是
中性
肯定
复制到用户范围。根据需要在分层中建立占位符目录（无文
634
识别出文本文件
632
将安装范围文件名称识别为候选文件名称，检查安装范围候选文件存在的类别(图12)
614
识别为文本文件
620
打开文件并返回结果
630
否定
否
将系统范围文件名称识别为候选文件名称
628
肯定
修改？
622
是
是
存在？
626
否
中性
文件未找到错误
624

图 8

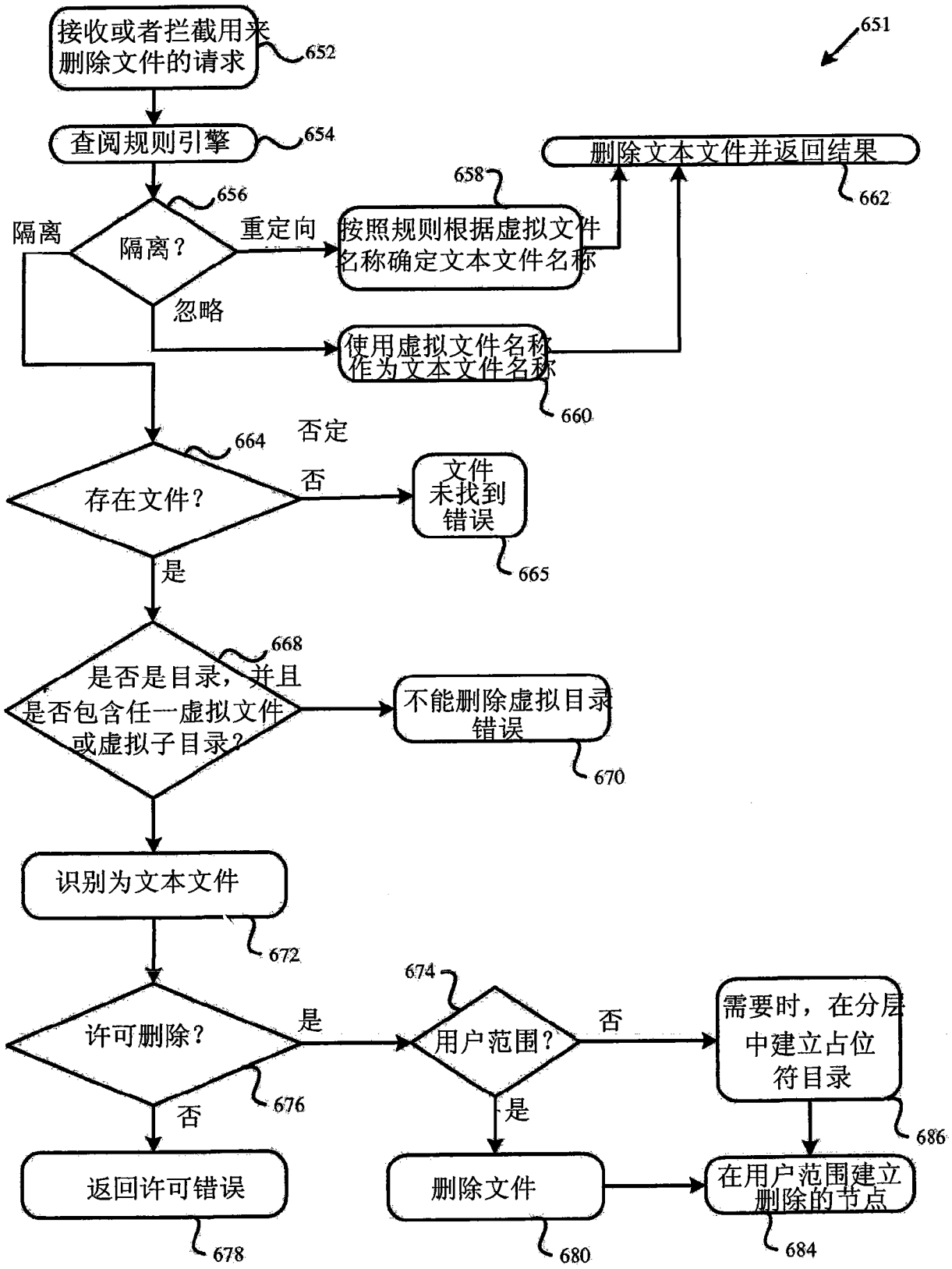

651
接收或者拦截用来删除文件的请求
652
查阅规则引擎
654
656
隔离？
隔离
重定向
忽略
658
按照规则根据虚拟文件名称确定文本文件名称
删除文本文件并返回结果
662
使用虚拟文件名称作为文本文件名称
660
664
存在文件？
否定
否
文件未找到错误
665
是
668
是否是目录，并且是否包含任一虚拟文件或虚拟子目录？
不能删除虚拟目录错误
670
识别为文本文件
672
许可删除？
是
否
676
返回许可错误
678
674
用户范围？
否
是
需要时，在分层中建立占位符目录
686
删除文件
680
在用户范围建立删除的节点
684

图 9

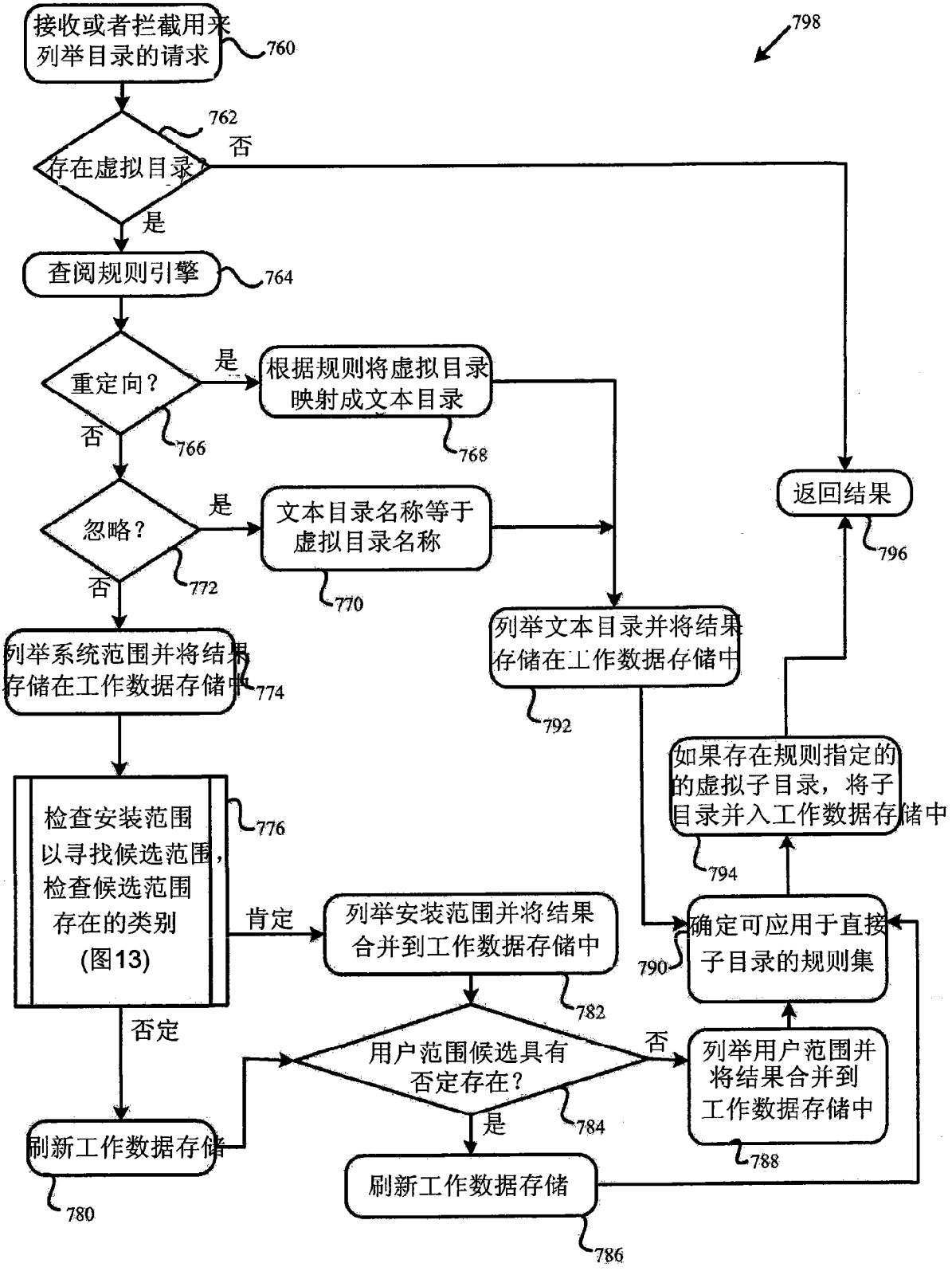

798
接收或者拦截用来列举目录的请求
760
762
存在虚拟目录？
否
是
查阅规则引擎
764
重定向？
是
根据规则将虚拟目录映射成文本目录
768
否
766
忽略？
是
文本目录名称等于虚拟目录名称
770
否
772
返回结果
796
列举系统范围并将结果存储在工作数据存储中
774
列举文本目录并将结果存储在工作数据存储中
792
如果存在规则指定的的虚拟子目录，将子目录并入工作数据存储中
794
检查安装范围以寻找候选范围，检查候选范围存在的类别(图13)
776
肯定
列举安装范围并将结果合并到工作数据存储中
782
确定可应用于直接子目录的规则集
790
否定
用户范围候选具有否定存在？
否
784
列举用户范围并将结果合并到工作数据存储中
788
是
刷新工作数据存储
780
刷新工作数据存储
786

图 10

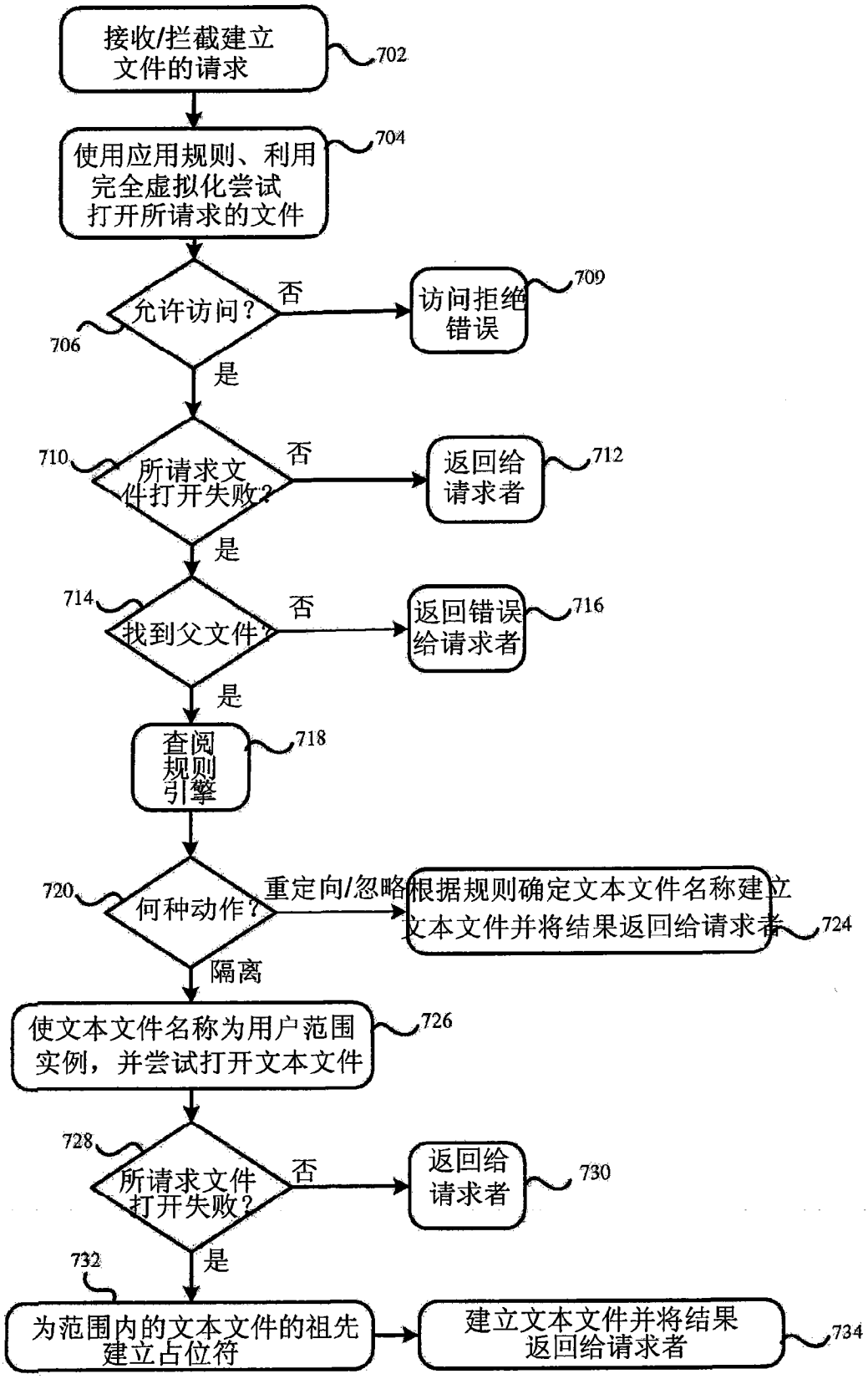

接收/拦截建立文件的请求
702
使用应用规则、利用完全虚拟化尝试打开所请求的文件
704
允许访问？
706
否
访问拒绝错误
709
是
所请求文件打开失败？
710
否
返回给请求者
712
是
找到父文件？
714
否
返回错误给请求者
716
是
查阅规则引擎
718
何种动作？
720
重定向/忽略
根据规则确定文本文件名称建立文本文件并将结果返回给请求者
724
隔离
使文本文件名称为用户范围实例，并尝试打开文本文件
726
所请求文件打开失败？
728
否
返回给请求者
730
是
为范围内的文本文件的祖先建立占位符
732
建立文本文件并将结果返回给请求者
734

图 11

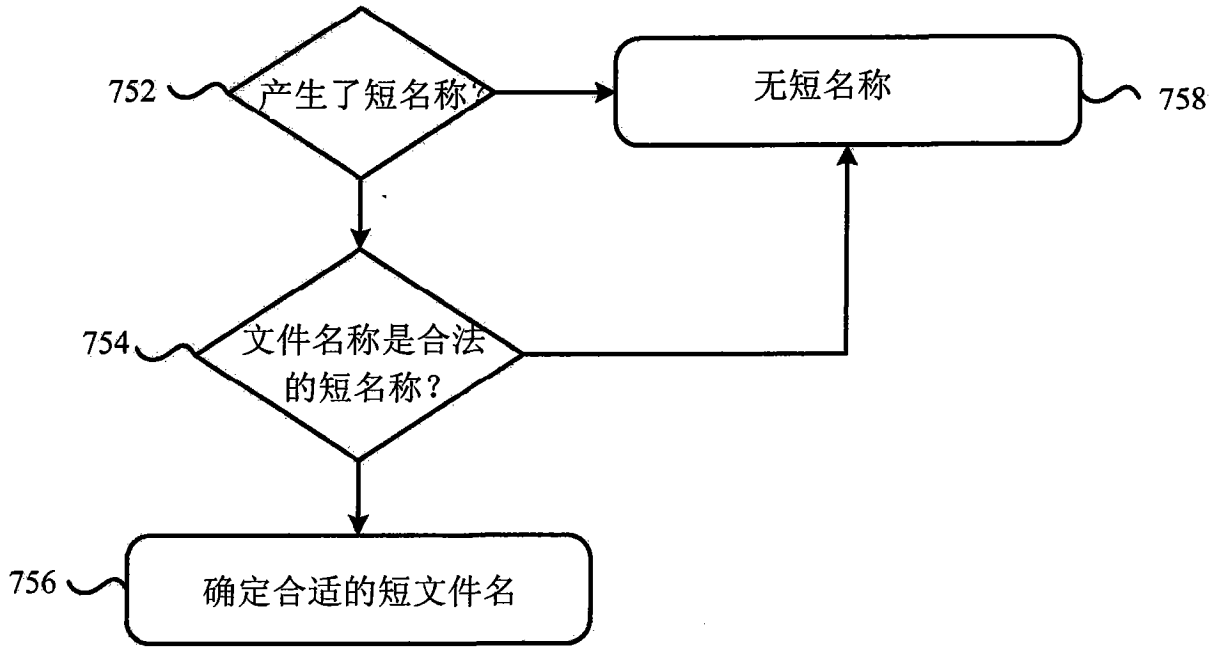


图 11A

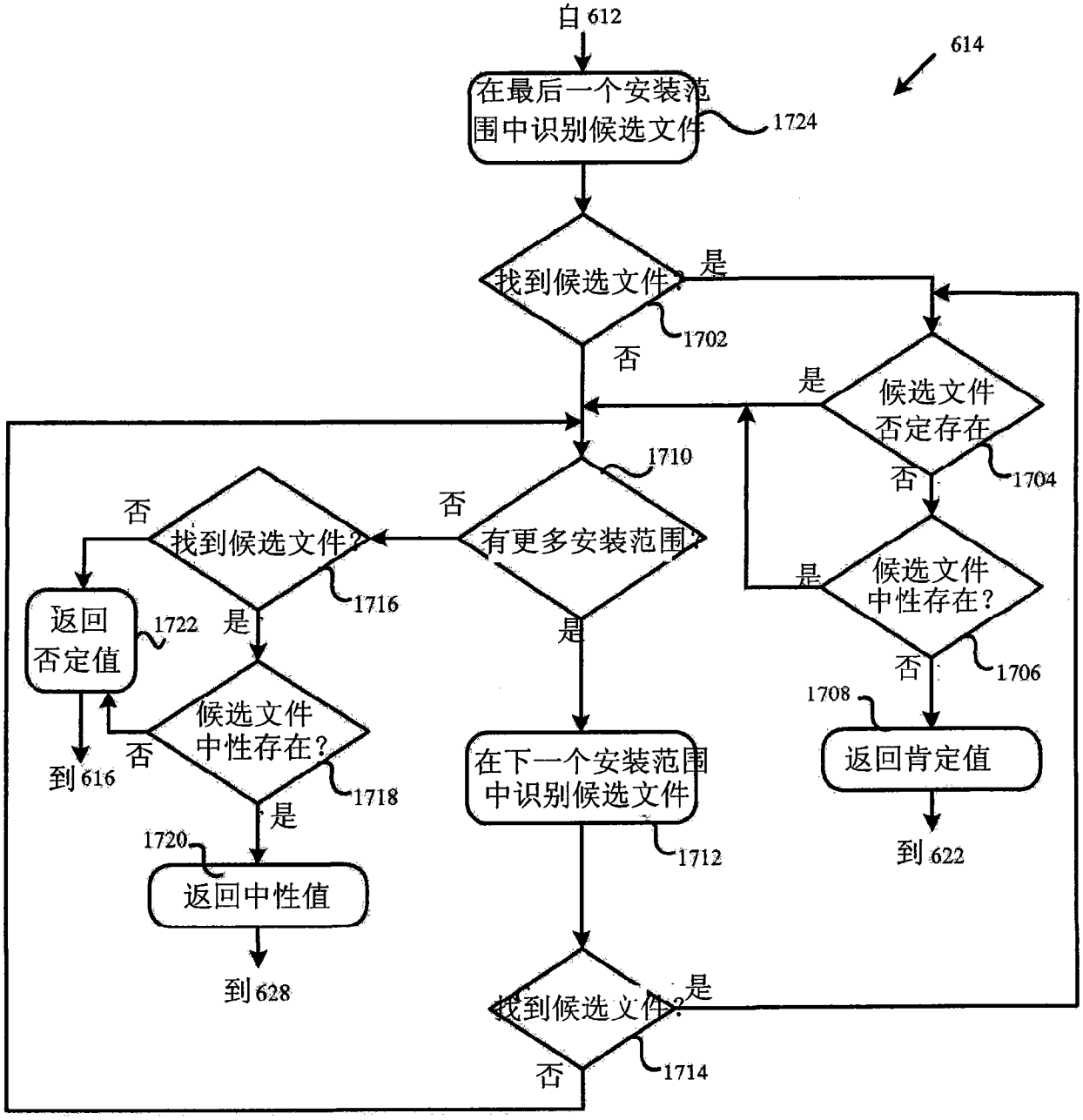

自 612
614
在最后一个安装范围中识别候选文件
1724
找到候选文件?
1702
是
否
候选文件否定存在
1704
是
否
候选文件中性存在?
1706
是
否
1708
返回肯定值
到 622
1710
有更多安装范围?
否
是
找到候选文件?
1716
否
是
返回否定值
1722
到 616
候选文件中性存在?
1718
否
是
1720
返回中性值
到 628
在下一个安装范围中识别候选文件
1712
找到候选文件?
1714
是
否

图 12

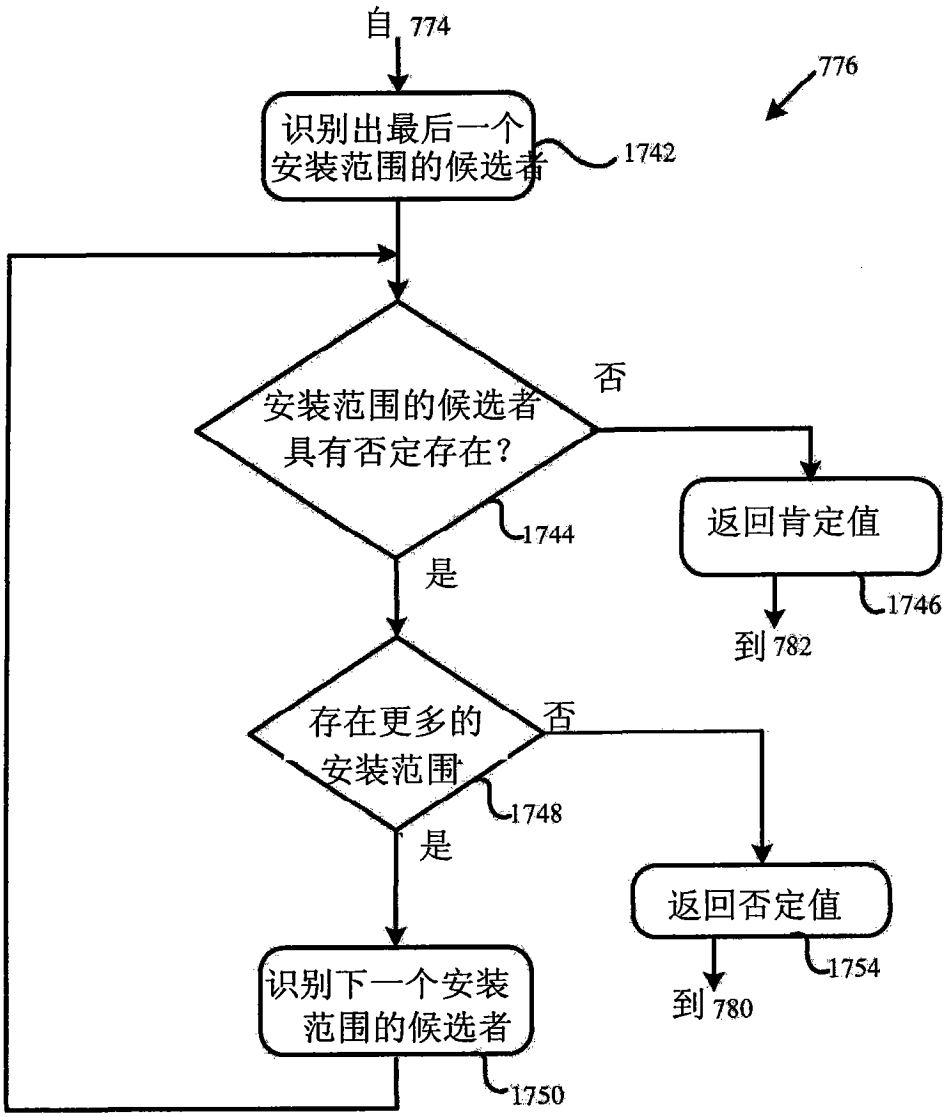

自 774
776
识别出最后一个安装范围的候选者
1742
安装范围的候选者具有否定存在？
否
1744
是
返回肯定值
1746
到 782
存在更多的安装范围
否
1748
是
返回否定值
1754
到 780
识别下一个安装范围的候选者
1750

图 13

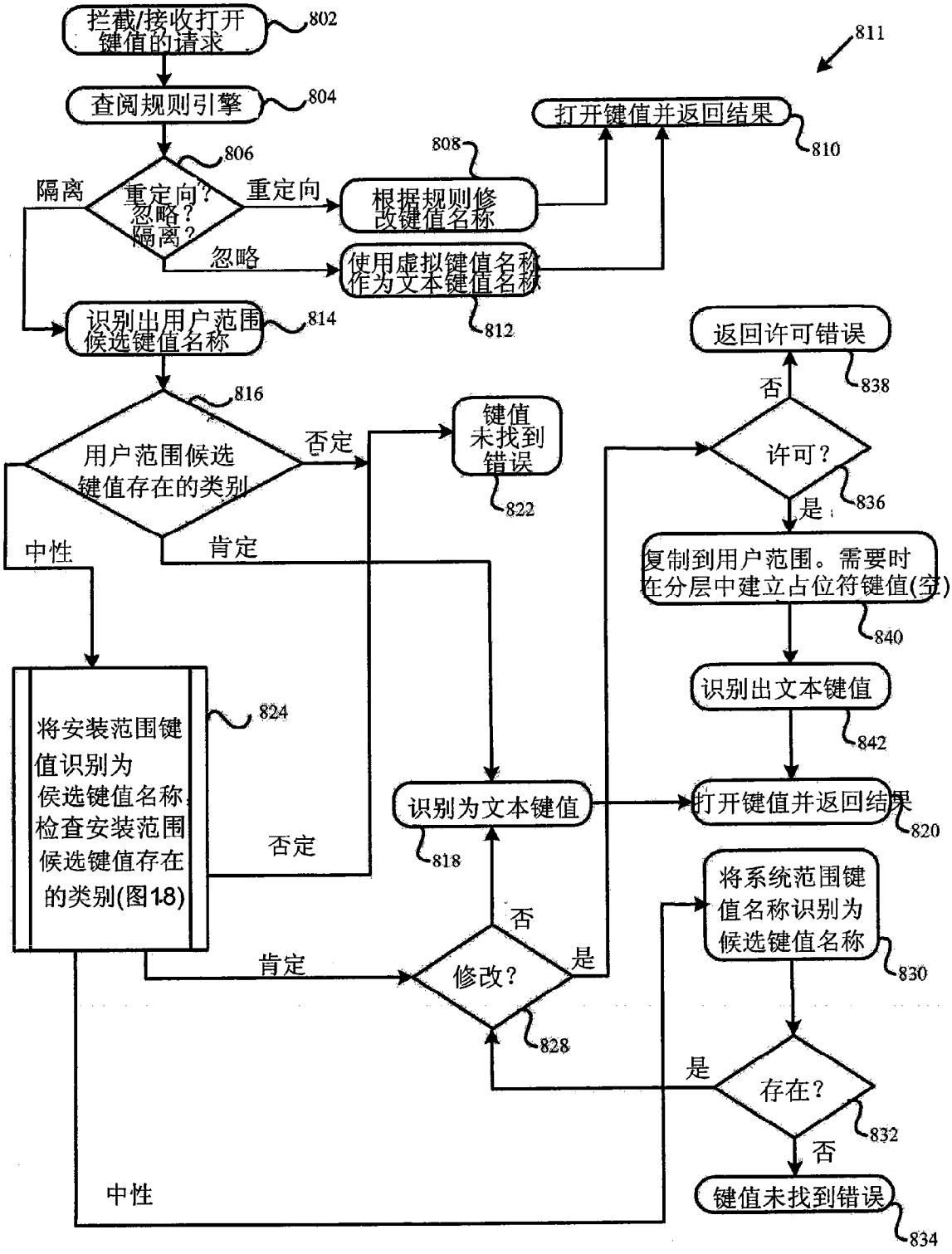

拦截/接收打开键值的请求
802
查阅规则引擎
804
806
重定向?
忽略?
隔离?
隔离
重定向
忽略
808
根据规则修改键值名称
使用虚拟键值名称作为文本键值名称
812
打开键值并返回结果
810
811
识别出用户范围候选键值名称
814
816
用户范围候选键值存在的类别
否定
键值未找到错误
822
中性
肯定
将安装范围键值识别为候选键值名称;检查安装范围候选键值存在的类别(图18)
824
否定
识别为文本键值
818
肯定
修改?
否
是
828
返回许可错误
838
否
许可?
836
是
复制到用户范围。需要时在分层中建立占位符键值(空)
840
识别出文本键值
842
打开键值并返回结果
820
将系统范围键值名称识别为候选键值名称
830
是
存在?
832
否
键值未找到错误
834
中性

图 14

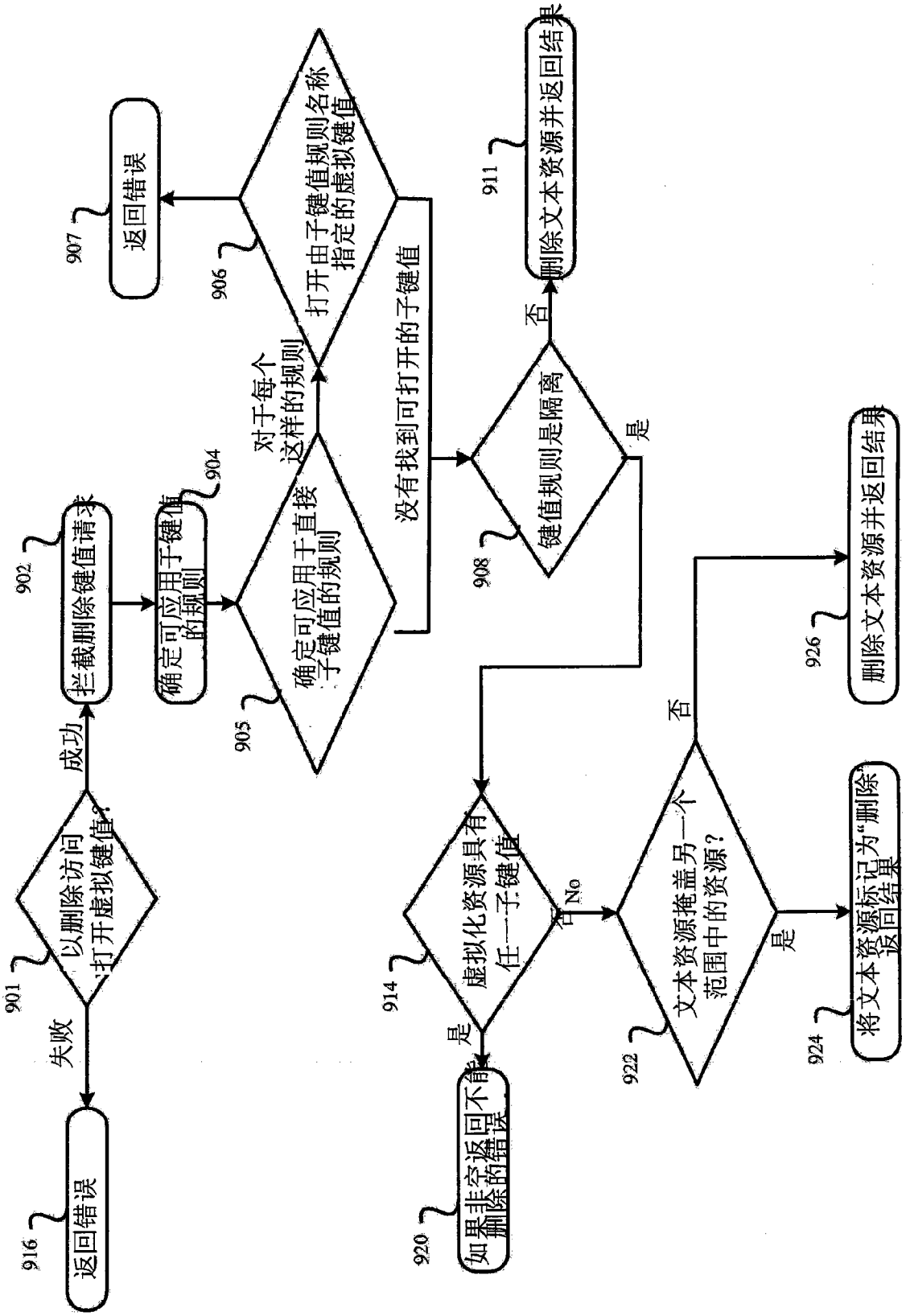

以删除访问打开虚拟键值？
901
失败
成功
拦截删除键值请求
902
确定可应用于键值的规则
904
确定可应用于直接子键值的规则
905
对于每个这样的规则
打开由子键值规则名称指定的虚拟键值
906
返回错误
907
没有找到可打开的子键值
键值规则是隔离
908
是
否
删除文本资源并返回结果
911
虚拟化资源具有任一子键值
914
是
否
No
如果非空返回不能删除的错误
920
文本资源掩盖另一个范围中的资源？
922
是
否
将文本资源标记为“删除”返回结果
924
删除文本资源并返回结果
926
返回错误
916

图 15

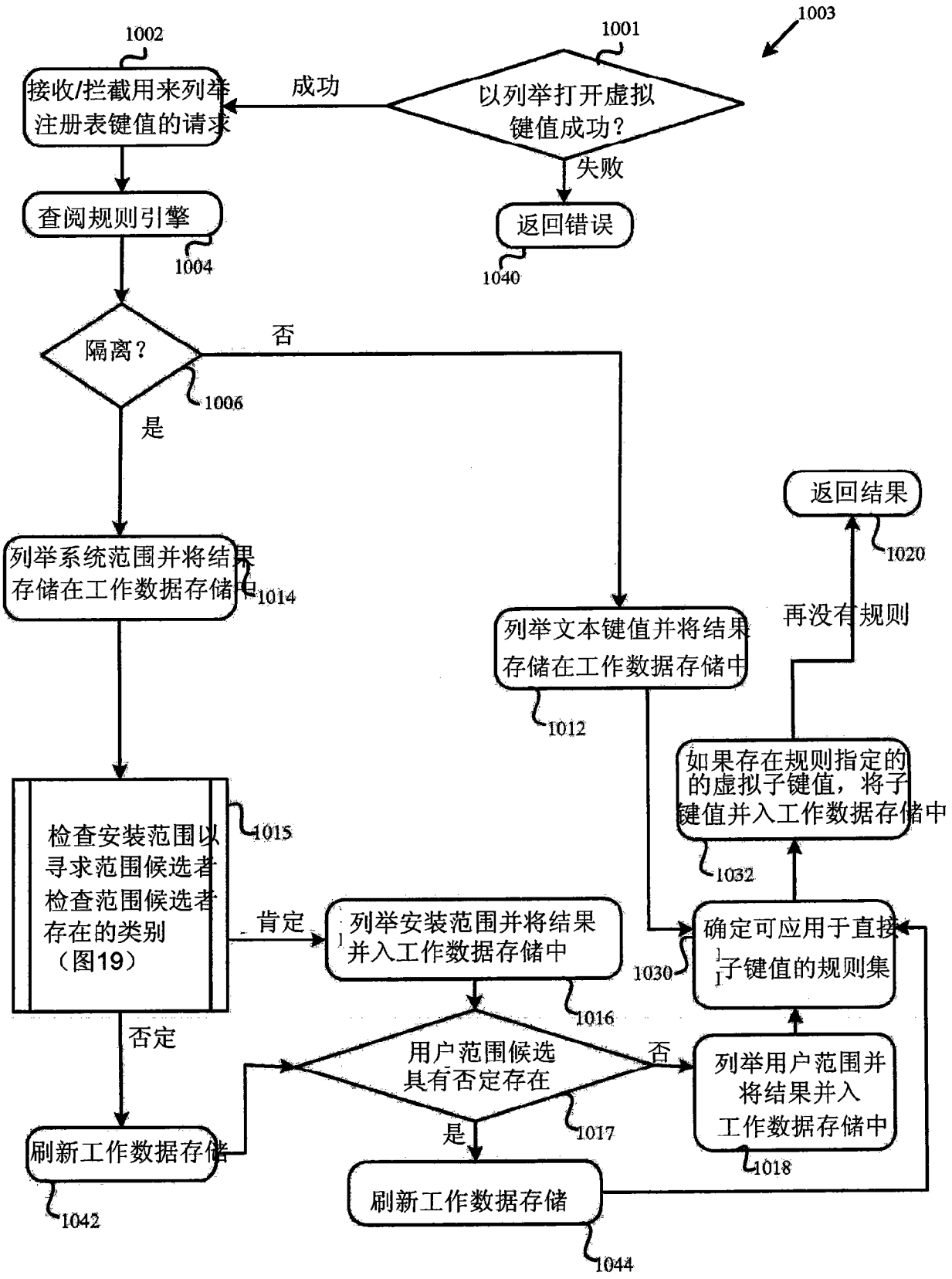

1003
1002
1001
接收/拦截用来列举注册表键值的请求
成功
以列举打开虚拟键值成功？
失败
查阅规则引擎
1004
返回错误
1040
隔离？
否
1006
是
返回结果
1020
列举系统范围并将结果存储在工作数据存储中
1014
列举文本键值并将结果存储在工作数据存储中
再没有规则
1012
如果存在规则指定的的虚拟子键值，将子键值并入工作数据存储中
1032
检查安装范围以寻求范围候选者检查范围候选者存在的类别（图19）
1015
肯定
列举安装范围并将结果并入工作数据存储中
确定可应用于直接子键值的规则集
1030
1016
否定
用户范围候选具有否定存在
否
列举用户范围并将结果并入工作数据存储中
1017
1018
是
刷新工作数据存储
刷新工作数据存储
1042
1044

图 16

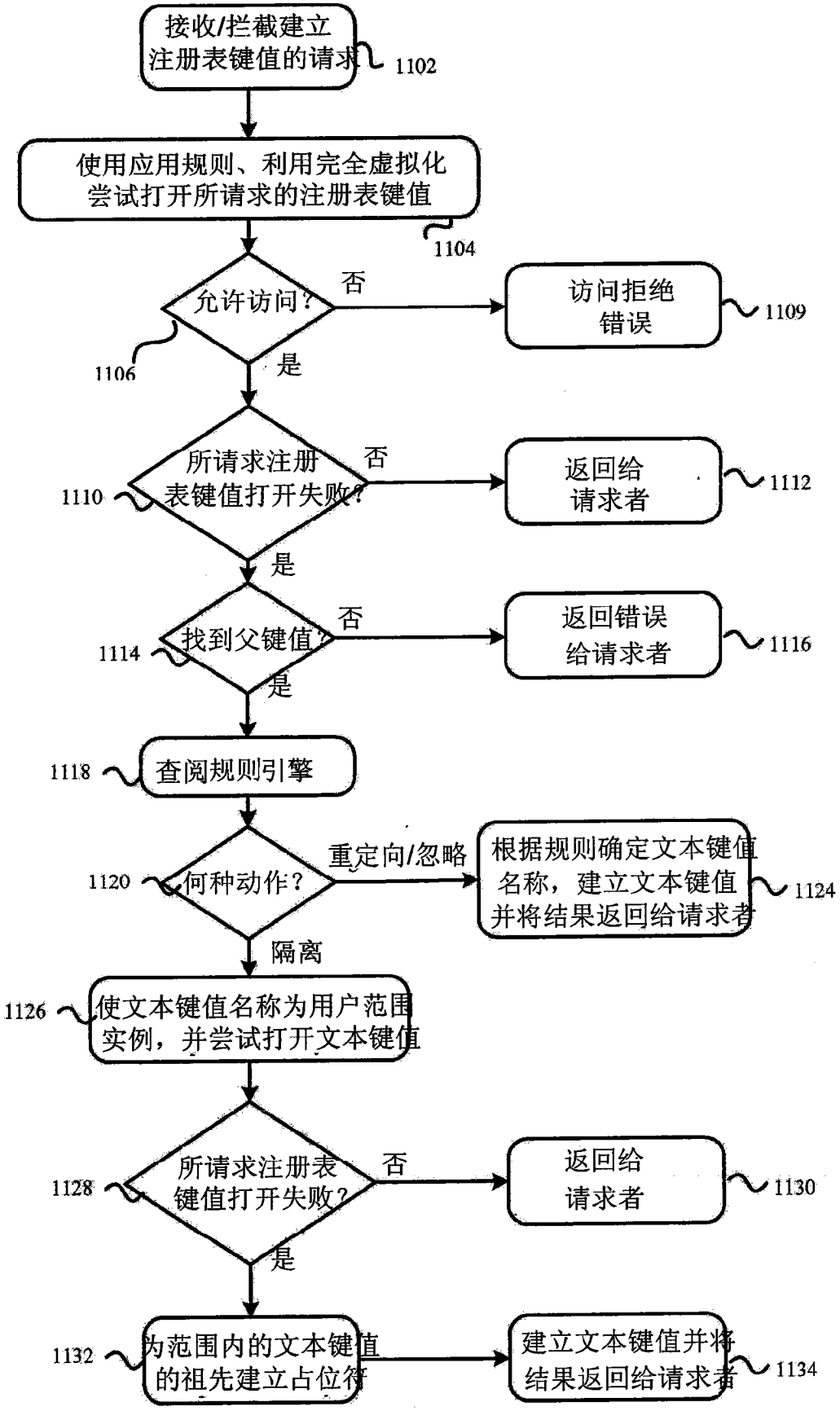

接收/拦截建立注册表键值的请求
1102
使用应用规则、利用完全虚拟化尝试打开所请求的注册表键值
1104
允许访问?
否
访问拒绝错误
1109
1106
是
所请求注册表键值打开失败?
1110
否
返回给请求者
1112
是
找到父键值?
1114
否
返回错误给请求者
1116
是
1118
查阅规则引擎
何种动作?
1120
重定向/忽略
根据规则确定文本键值名称，建立文本键值并将结果返回给请求者
1124
隔离
1126
使文本键值名称为用户范围实例，并尝试打开文本键值
所请求注册表键值打开失败?
1128
否
返回给请求者
1130
是
1132
为范围内的文本键值的祖先建立占位符
建立文本键值并将结果返回给请求者
1134

图 17

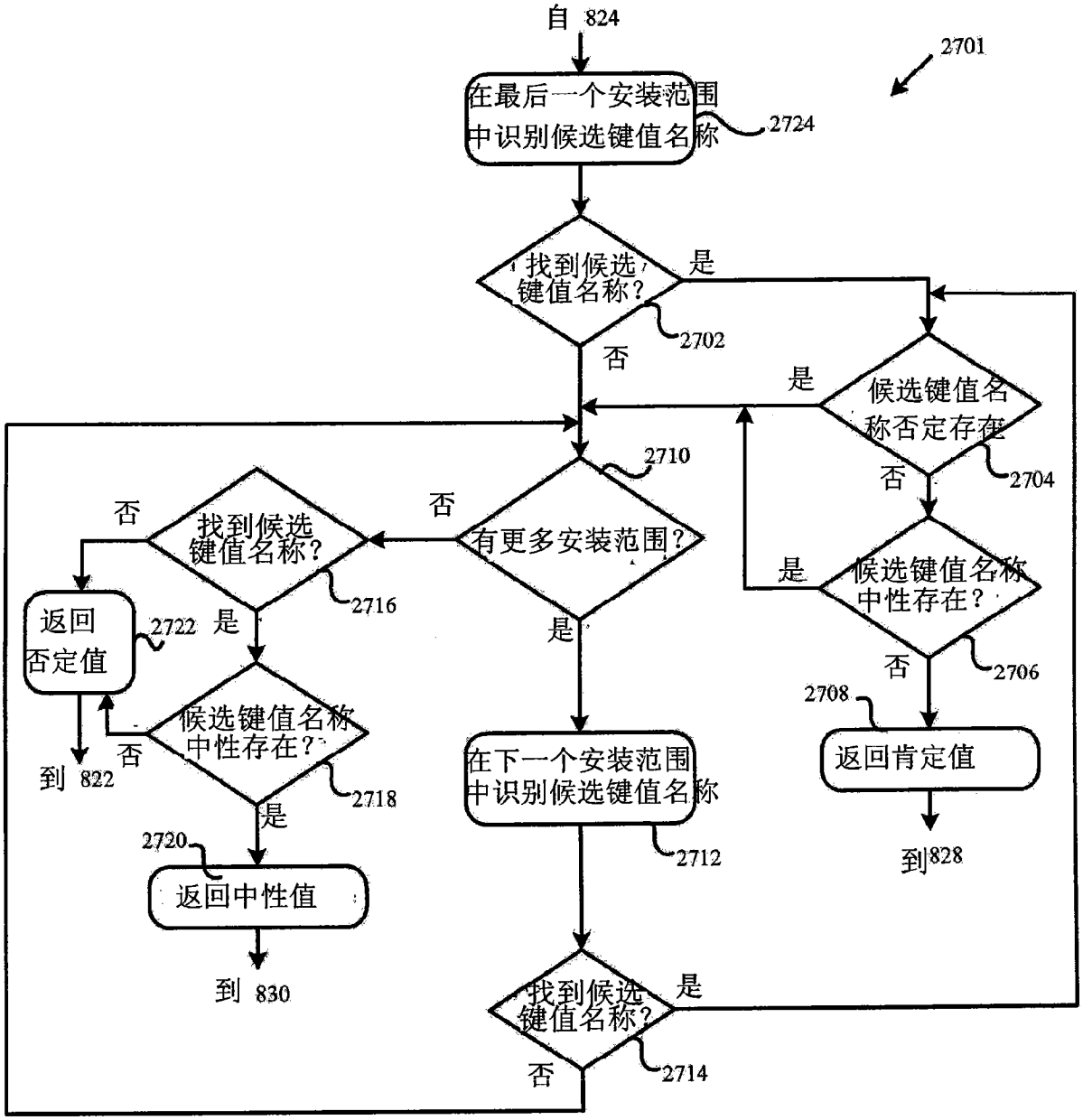

自 824
2701
在最后一个安装范围中识别候选键值名称
2724
找到候选键值名称?
是
2702
否
候选键值名称否定存在?
是
2704
否
2710
有更多安装范围?
否
找到候选键值名称?
否
候选键值名称中性存在?
是
2706
2708
返回肯定值
到828
返回否定值
2722
是
2716
候选键值名称中性存在?
否
到 822
2718
是
2720
返回中性值
到 830
在下一个安装范围中识别候选键值名称
2712
找到候选键值名称?
是
2714
否

图 18

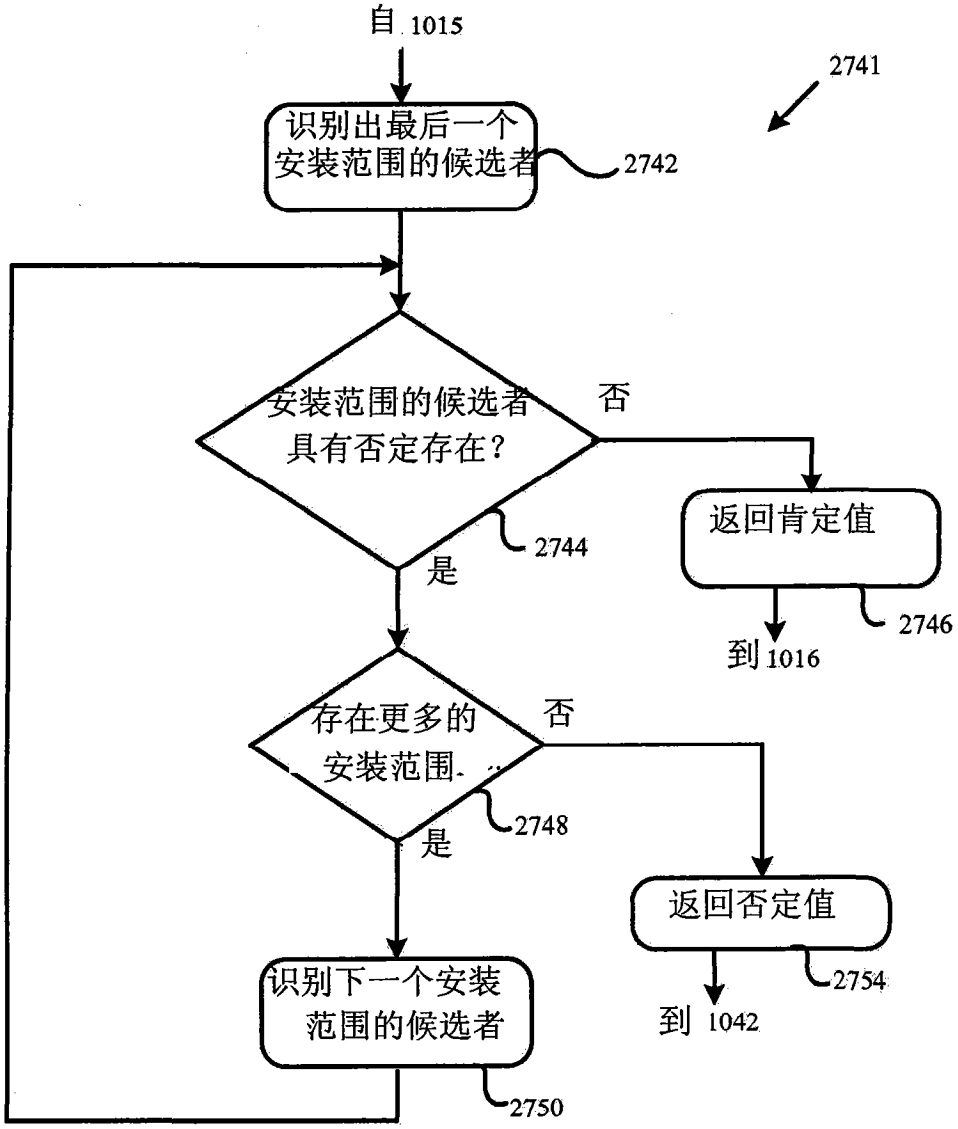

自 1015
2741
识别出最后一个安装范围的候选者
2742
安装范围的候选者具有否定存在？
否
2744
是
返回肯定值
2746
到 1016
存在更多的安装范围
否
2748
是
返回否定值
2754
到 1042
识别下一个安装范围的候选者
2750

图 19

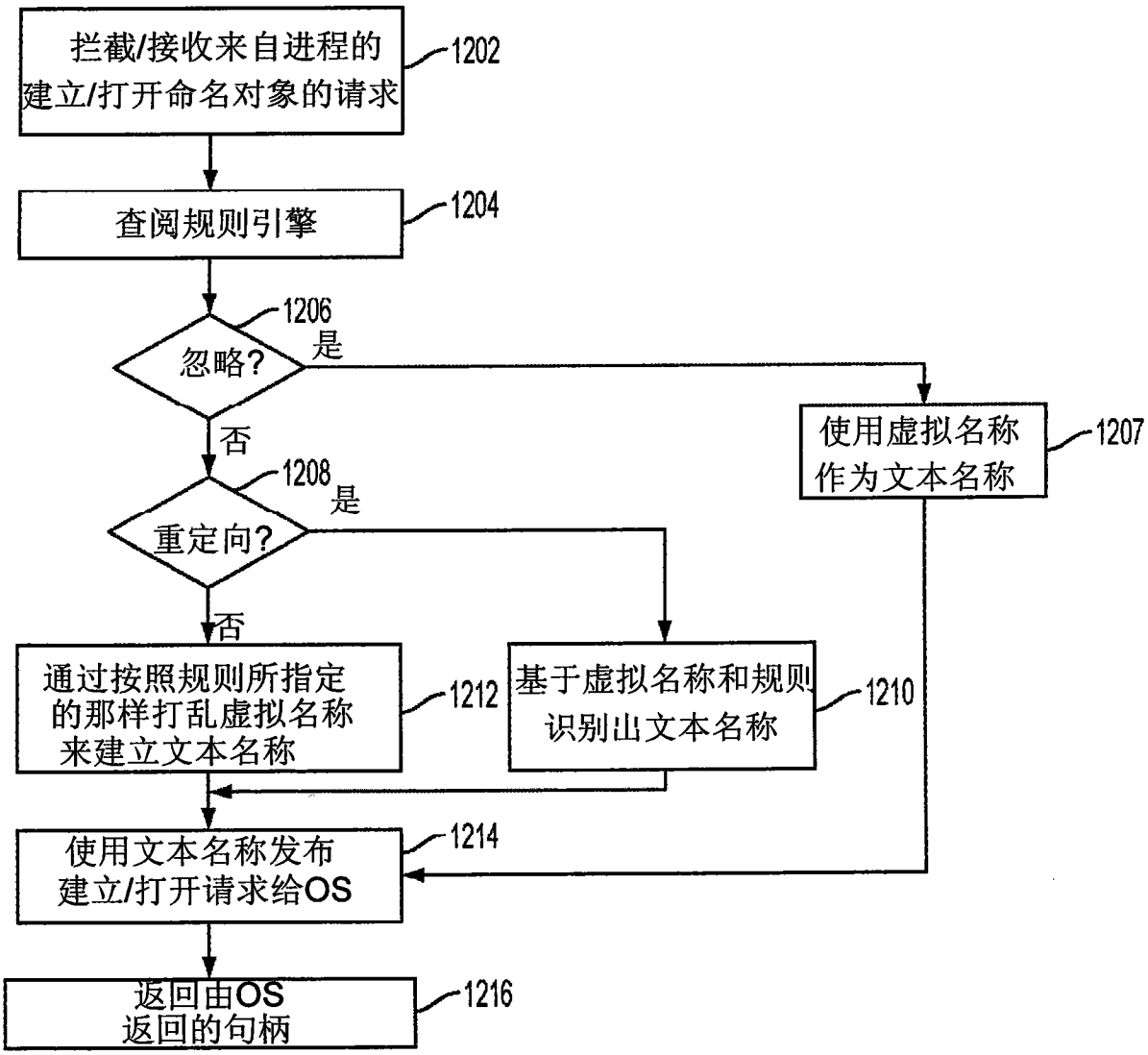

拦截/接收来自进程的建立/打开命名对象的请求
1202
查阅规则引擎
1204
1206
忽略?
是
否
使用虚拟名称作为文本名称
1207
1208
重定向?
是
否
通过按照规则所指定的那样打乱虚拟名称来建立文本名称
1212
基于虚拟名称和规则识别出文本名称
1210
使用文本名称发布建立/打开请求给OS
1214
返回由OS返回的句柄
1216

图 20

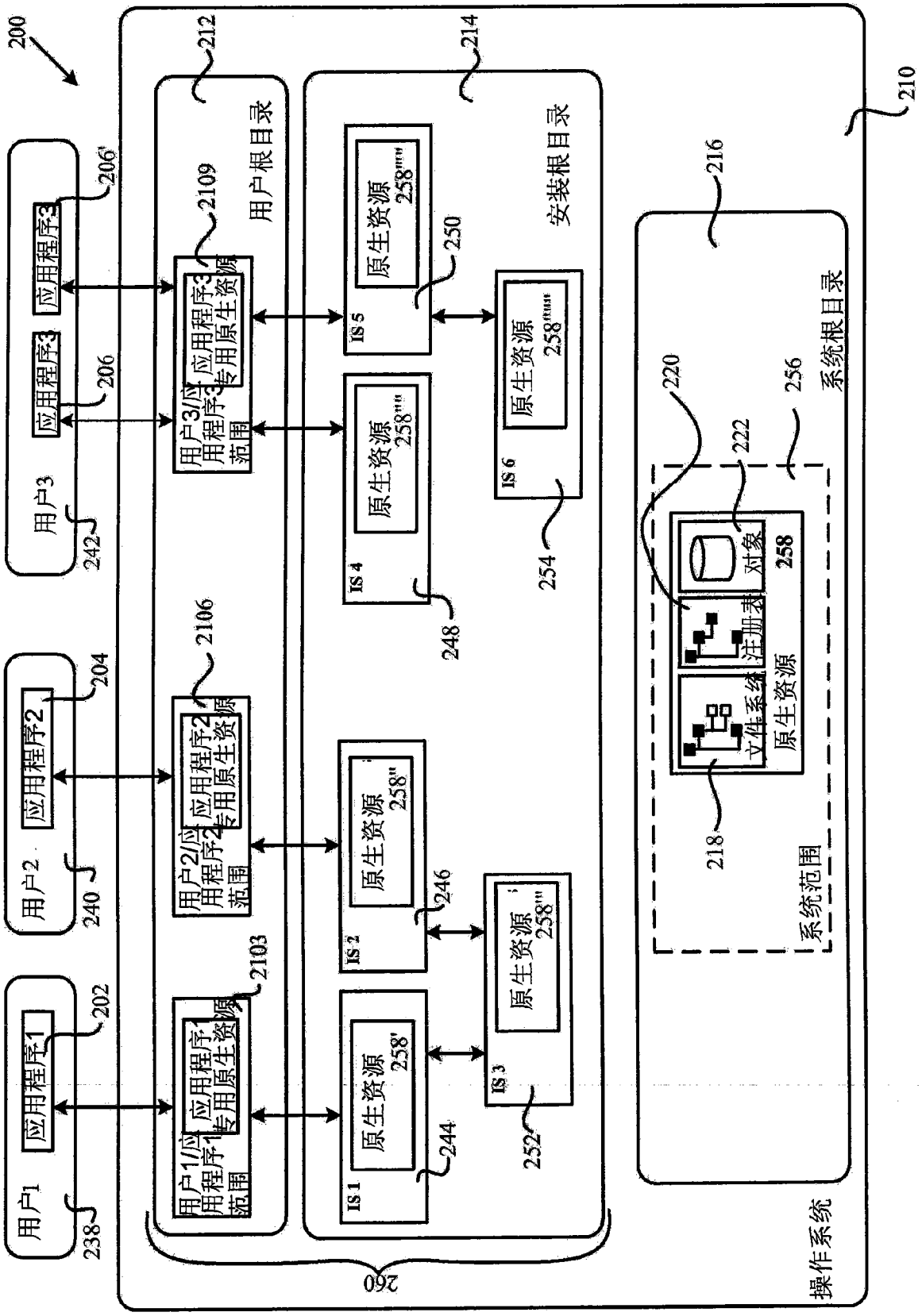

200
210
212
214
216
用户根目录
安装根目录
系统根目录
用户1
用户2
用户3
应用程序1
应用程序2
应用程序3
202
204
206
206'
238
240
242
用户1/应用程序1专用原生资源范围
用户2/应用程序2专用原生资源范围
用户3/应用程序3专用原生资源范围
2103
2106
2109
IS 1
IS 2
IS 3
IS 4
IS 5
IS 6
原生资源 258'
原生资源 258"
原生资源 258""
原生资源 258"""
原生资源 258""""
244
246
248
250
252
254
256
258
220
222
218
文件系统
注册表
对象
原生资源
系统范围
260
操作系统

图 21A

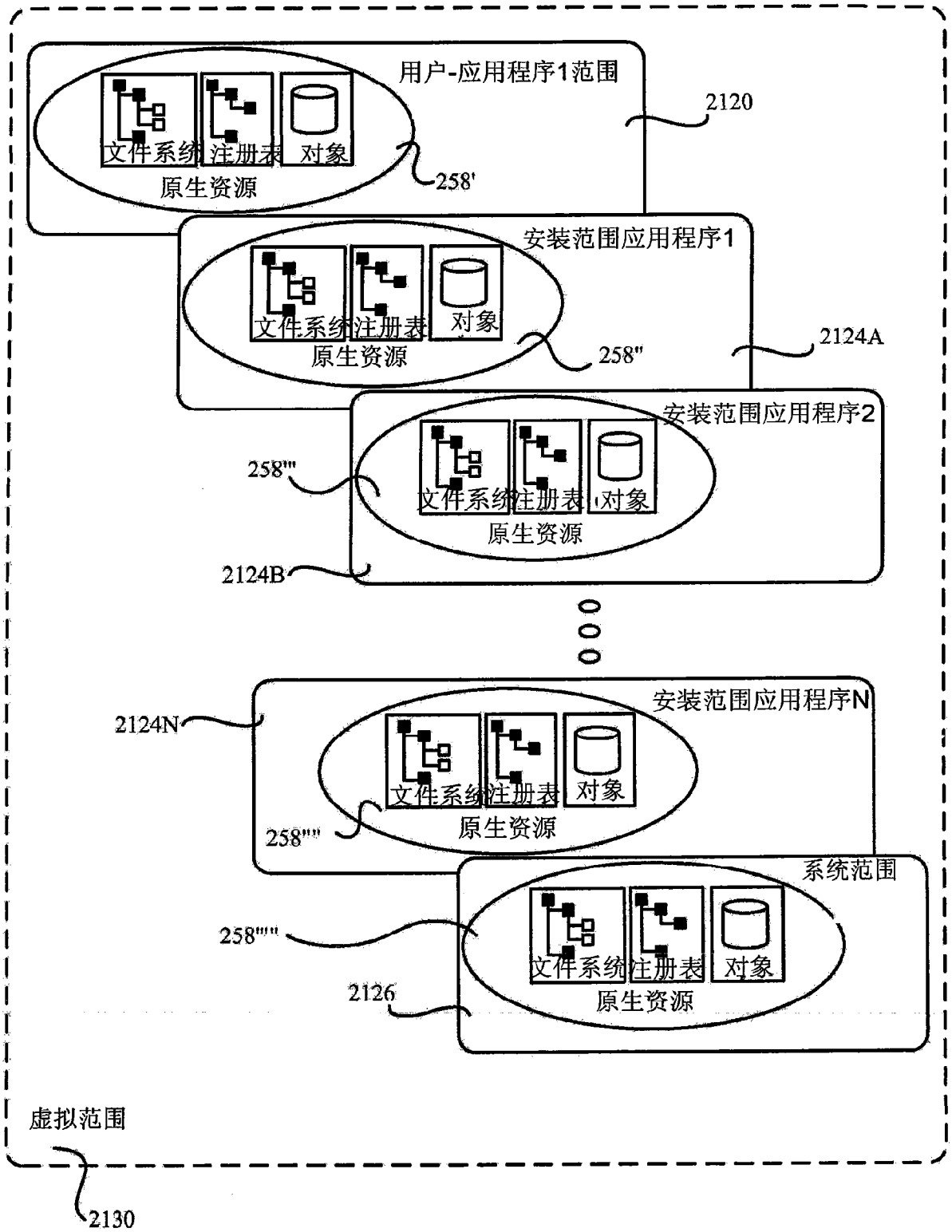

用户-应用程序1范围
2120
文件系统
注册表
对象
原生资源
258'
安装范围应用程序1
文件系统
注册表
对象
原生资源
258"
2124A
安装范围应用程序2
258'''
文件系统
注册表
对象
原生资源
2124B
安装范围应用程序N
2124N
文件系统
注册表
对象
原生资源
258''''
系统范围
258'''''
文件系统
注册表
对象
原生资源
2126
虚拟范围
2130

图 21B

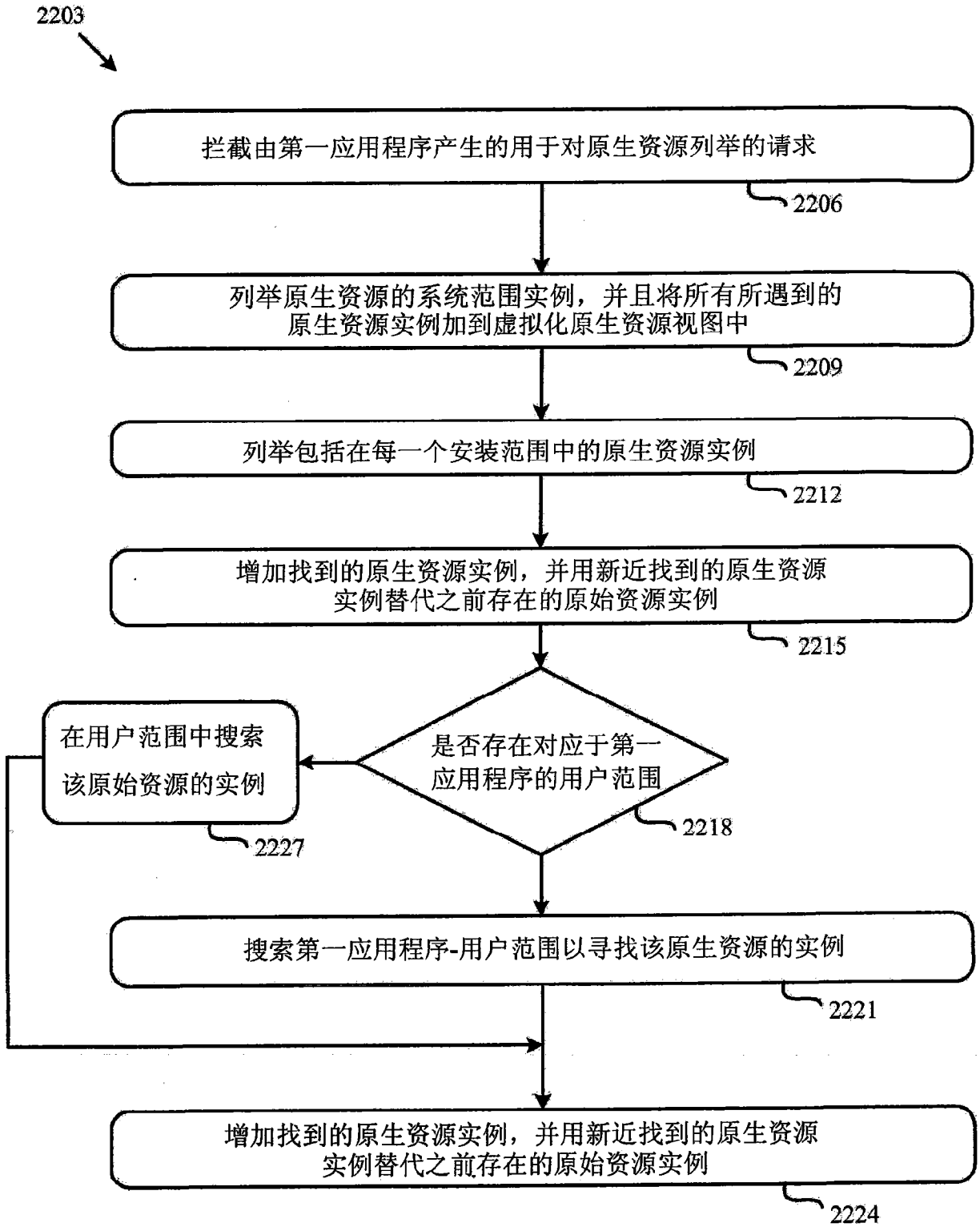

2203
拦截由第一应用程序产生的用于对原生资源列举的请求
2206
列举原生资源的系统范围实例，并且将所有所遇到的原生资源实例加到虚拟化原生资源视图中
2209
列举包括在每一个安装范围中的原生资源实例
2212
增加找到的原生资源实例，并用新近找到的原生资源实例替代之前存在的原始资源实例
2215
是否存在对应于第一应用程序的用户范围
2218
在用户范围中搜索该原始资源的实例
2227
搜索第一应用程序-用户范围以寻找该原生资源的实例
2221
增加找到的原生资源实例，并用新近找到的原生资源实例替代之前存在的原始资源实例
2224

图 21C