[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)
G06F 9/44 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410011929.0

[45] 授权公告日 2009 年 7 月 1 日

[11] 授权公告号 CN 100507904C

[22] 申请日 2004.9.21
[21] 申请号 200410011929.0
[30] 优先权
[32] 2003.10.24 [33] US [31] 60/514,069
[32] 2004.7.30 [33] US [31] 10/909,217
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 A·W·坎特 D·J·德索扎
M·霍斯特曼 M·J·格里尔
S·G·希诺 S·帕沙沙拉西
[56] 参考文献
WO01/98926A2 2001.12.27
Apache Ant 1.5 Manual. Apache Software Foundation. 2002
Building Eclipse Instance Messenger. Zhong Chen. http://www.scs.carleton.ca/~arpwhite/documents/honoursProjects/paul.chen.2003.pdf. 2003
Building and Deploying applications with ANT. Gendelman Michael. BEA. 2002
JAR File Specification. http://web.archive.org/web/20030607203613/http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html. 2003
审查员 王艳臣
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 谢喜堂

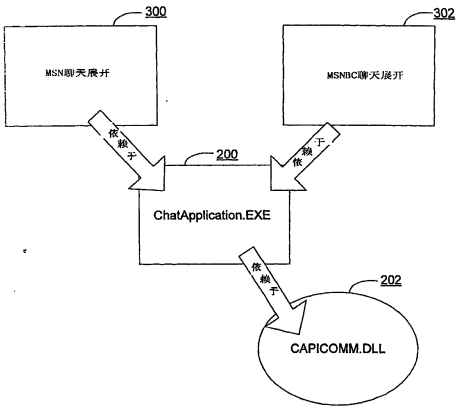
权利要求书 2 页 说明书 10 页 附图 3 页

[54] 发明名称

构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架

[57] 摘要

揭示了一种构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架。该框架被声明性地定义为处理身份，即强健身份的清单。该应用程序清单可声明安全地配置或定制应用程序的适当的方法，并提供仅向授权的多方授予这一权限的能力。本发明的另一方面提供了一种用于采用处理定制应用程序的身份的清单声明性地定义的应用程序展开的框架。该框架为系统、状态基础结构、设置程序、创作工具和管理工具提供了一种方法以使用授权的复合应用程序身份展开、安装、服务并管理定制应用程序。应用程序清单以及展开清单可在展开的应用程序的整个生命周期内－包括运行时－令其变得可用，从而帮助了定制应用程序的一致操作。



1．一种用于安全地定制应用程序的方法，其特征在于，所述方法包括：

确定所述定制应用程序运行的上下文环境；

从所述运行定制应用程序的上下文环境内获取应用程序身份；

根据所述应用程序身份获取对应于所述应用程序的一展开清单文件，所述展开清单文件定义了该应用程序身份、一组定制的展开属性和一组应用程序的属性；

根据所获得的展开清单文件中该组定制的展开属性和该组应用程序的属性，从所述展开清单文件确定可用的展开选项；以及

响应于所述确定可用的展开选项，执行许可的应用程序定制活动。

2．如权利要求1所述的方法，其特征在于，所述确定可用的展开选项包括确定许可的定制活动。

3．如权利要求1或2所述的方法，其特征在于，所述确定可用的展开选项包括确定被授权来执行应用程序定制的多个被授权方。

4．如权利要求1所述的方法，其特征在于，所述获取一展开清单文件包括从在其上储存了与定制的应用程序关联的应用程序清单模式数据结构的计算机可读媒质处获得展开清单文件，其中，所述数据结构包括：

第一数据字段，它包含表示指示所述模式包含应用程序清单信息的元素的数据；以及第二数据字段，它包含表示应用程序标识符的数据。

5．如权利要求4所述的方法，其特征在于，所述第二数据字段的中的应用程序标识符包括与定制应用程序关联的应用程序标识符数据结构，其中，所述应用程序标识符数据结构包括：第一数据字段，它包含表示一个或多个展开清单数据结构的数据；以及第二数据字段，它包含表示应用程序清单数据结构的数据。

6．如权利要求1所述的方法，其特征在于，所述获取一展开清单文件包括从在其上储存了与定制的应用程序关联的应用程序清单模式数据结构的计算机可读媒质处获得展开清单文件，其中，所述数据结构包括：第一数据字段，它包含指示所述模式包含展开清单信息的元素的数据；以及第二数据字段，它包含表示定制应用程序标识符的数据。

7．如权利要求1所述的方法，其特征在于，所述获取应用程序身份的步骤进一步包括：

从所述应用程序身份中获取展开身份，所述展开身份单独地标识所述定制。

8．如权利要求 1 所述的方法，其特征在于，所述应用程序身份包括通过原始应用程序的依赖图的路径并被表示为沿该路径的组件的组件身份的排序列表；并且

所述方法进一步包括：

使用所述应用程序身份将政策状态的作用域限定至一个或多个应用程序，其中所述限定包括绑定应用程序身份的每个部分并确定该组件是否与服务器语义同步。

9．如权利要求 1 所述的方法，其特征在于，所述获取应用程序身份的步骤进一步包括：

调用一应用程序管理应用编程接口；以及

从所述应用编程接口获取用于所安装的应用程序的应用程序身份，所述所安装的应用程序包括原始应用程序和其展开，其中所述所安装的应用程序的应用程序身份包括下述全身份中的一个或多个：展开者的信息，公钥令牌、版本信息和处理器体系信息；

其中所述原始应用程序的公钥令牌和展开是相同的。

10．如权利要求 1 所述的方法，其特征在于，所述方法进一步包括通过下述步骤构建所述展开清单文件：

定义一包括所有构成组件的内容的展开清单数据结构；

将所述展开清单与所述应用程序关联；以及

在所述展开清单内指定定制数据以形成合并的应用程序清单，所述合并应用程序清单是展开清单数据结构的编译，而不考虑任一组件绑定政策并且不考虑任一外部或先决依赖。

11．如权利要求 10 所述的方法，其特征在于，所述应用程序的比特被维持不变。

构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架

技术领域

本发明一般涉及计算机系统软件应用程序，尤其涉及可定制和可配置可重复使用应用程序。

相关申请的参照

本申请要求2003年10月24日提交的美国临时申请序列号60/514,069的优先权。

背景技术

当今的软件应用程序通常允许某种程度的定制和/或配置。定制可发生在应用程序从制造商处发货之前。例如，应用程序的本地化版本（如，英语、德语、日语等）可由软件开发者在发行并分发之前创建。定制也可发生在用户安装了该产品之后，如服务包或其它软件更新的情况。用户一般也可以在安装的软件内配置设定和选项，如色彩和声音模式、图标、启动屏幕等等。

应用程序的展开者（deployer）通常需要定制并配置他们所展开的应用程序。例如，信息技术管理员可能需要将其网络域内使用的应用程序的外观和感觉和/或特征设定标准化。在另一示例中，因特网服务提供商可能期望向其订户分发应用程序的有商标的版本。常规地，这一定制通过修改应用程序的安装过程（setup）或在其安装过程运行之后使用另一自定义程序修改所展开的应用程序的状态来完成。尽管机器的结果状态事实上可能令应用程序处于其定制形式，然而没有授权的定义或句炳来操纵这一定制的应用程序。

上述情形是有问题的，因为一旦定制了应用程序，该应用程序要么丢失了其原始的应用程序身份并拥有不相关的新身份，要么仍具有与定制的身份相对的原始身份，，没有一种方法来将定制的应用程序标识为原始应用程序的定制变异体。由此，确保在应用程序的整个生命周期中（即，安装、定制、更新、执行和收回应用程序）它能够被服务并在必要时发挥作用一依照原始应用程序，变得尤其具有挑战

性。

发明内容

鉴于上述情况，本发明提供了一种构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架。具体地，本发明提出一种用于要被声明性地定义为拥有身份，包括一但不必是一强身份（见 2000 年 6 月 28 日提交的名为“共享名字（Shared Names）”的申请号 09/605,602，其内容通过引用整体结合于此）的清单的应用程序的框架（即，一种控制其执行环境并可以被激活的组件）。该应用程序清单能够声明适当的方法来安全地配置或定制该应用程序，并提供仅向授权方授予这一权限的能力。

本发明的另一方面是，它也提供了一种用于要被声明性地定义为拥有定制的应用程序的身份的清单的应用程序展开的框架。这一框架向系统、状态基础结构、设置程序、授权工具、管理工具和其它感兴趣的各方提供了一种方法来使用授权复合应用程序身份展开、安装、服务和管理定制的应用程序。该定制的应用程序也可以被进一步定制。

应用程序清单以及展开清单可以在所展开的应用程序的整个生命周期中一包括运行时一令其变得可用，这帮助了定制的应用程序的一致操纵。对允许多个这样的定制的应用程序在同一作用域内（即，用户、机器、网络等等）可用的支持可以得到真正的可重复使用的应用程序，与并排组件十分相似。

附图说明

尽管所附权利要求书用细节陈述了本发明的特征，然而结合附图阅读以下详细描述可以最好地理解本发明及其对象和优点，附图中：

图 1 是可在其中实现本发明的框架的示例性计算机体系结构的示意图；

图 2 是可在其中实现本发明的框架的示例性软件应用程序的示意图；

图 3 是可在其中实现本发明的示例性展开的示意图。

具体实施方式

在以下描述中，参考由一个或多个计算机执行的行动和操作的符号表示来描述本发明，除非另外指明。由此，可以理解，这些行动和操作，不时被称为计算机

可执行的，包括由计算机的处理单元对表示结构化形式的数据的电信号的操纵。这一操纵对数据进行转换或在计算机的存储器系统中的位置上维护它们，从而以本领域的技术人员都理解的方式重新配置或改变计算机的操作。维护着数据的数据结构是具有由该数据的格式所定义的具体特征的存储器的物理位置。然而，尽管在上述上下文语境中描述本发明，它并不意味着局限，如本领域的技术人员所理解的，后文所描述的若干行动和操作也能以硬件实现。

转向附图，其中，相同的标号标识相同的元件，示出了本发明在合适的计算环境内的实现。以下描述基于本发明的说明性实施例，不应当被视为是就此处未明确描述的替换性实施例而局限本发明。

示例性环境

参考图 1，本发明涉及连接的计算机网络上的网络节点之间的通信。每一网络节点驻留在具有许多不同的计算机体系结构之一的计算机内。为描述目的，图 1 示出了可用于这些装置的示例性计算机体系结构的示意图。所描绘的体系结构仅是合适的环境的一个示例，并非局限本发明的使用或功能的范围。也不应将该计算装置解释为对图 1 所示的任一组件或其组合具有依赖或需求。本发明可使用众多其它通用或专用计算或通行环境或配置来操作。适合使用本发明的众所周知的计算系统、环境和配置包括但不限于，移动电话、袖珍计算机、个人计算机、服务器、多处理器系统、基于微处理器的系统、小型机、大型机和包括上述系统或装置的任一个的分布式计算环境。

在其最基本的配置中，计算装置 100 通常包括至少一个处理单元 102 和存储器 104。存储器 104 可以是易失的（如 RAM）、非易失的（如 ROM 和闪存）或两者的某一组合。该最基本的配置在图 1 中由虚线 106 示出。

计算装置 100 也可包含可具有另外的特征和功能的存储媒质设备 108 和 110。例如，它们可包括另外的存储（可移动和不可移动），包括但不限于，PCMCIA 卡、磁和光盘和磁带。这类另外的存储在图 1 中由可移动存储 108 和不可移动存储 110 示出。计算机存储媒质包括以用于储存信息的任一方法或技术实现的易失和非易失、可移动和不可移动媒质，信息如计算机可读指令、数据结构、程序模块或其它数据。存储器 104、可移动存储 108 和不可移动存储 110 都是计算机存储媒质的示例。计算机存储媒质包括但不限于，RAM、ROM、EEPROM、闪存、其它存储

器技术、CD-ROM、数字多功能盘、其它光存储、磁盒、磁带、磁盘存储、其它磁存储设备以及可用于储存所期望的信息并可由计算机装置访问的任一其它媒质。

计算装置 100 也可包含允许其与其它装置进行通信的通信信道 112。通信信道 112 是通信媒质的示例。通信媒质通常在诸如载波或其它传输机制等已调制数据信号中实施计算机可读指令、数据结构、程序模块或其它数据，并包括任一信息传送媒质。术语“已调制数据信号”指以对信号中的信息进行编码的方式设置或改变其一个或多个特征的信号。作为示例而非局限，通信媒质包括有线媒质，如有线网络和直接连线连接，以及无线媒质，如声学、无线电、红外和其它无线媒质。本发明使用的术语计算机可读媒质包括存储媒质和通信媒质。计算装置 100 也具有输入组件 114，如键盘、鼠标、输入笔、语音输入组件和触摸输入设备。输出组件 116 包括屏幕显示器、扬声器、打印机和用于驱动它们的呈现模块（通常称为“适配器”）。计算装置 100 具有电源 118。所有这些组件在本领域中是众所周知的，不需要在这里详细描述。

构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架

本发明针对一种构建、展开、服务并管理可定制和可配置可重复使用应用程序的框架。简言之，应用程序是一组如由主机所准许的控制其执行环境并可被激活的一个或多个组件。组件可以被描述为原子性、不可变文件组。应当注意，本发明所使用的“控制”是相对性的；有某些甚至是应用程序也不能控制的因素（如，整个机器范围内的组件服务/设置等等）。同样，有提供不同的保证和控制级别以及对应用的隔离的不同级别的环境（如，机器、进程、应用域（appdomain））。另外，应用程序不必是可执行文件，并且可以不运行其自己的进程，而相反运行在一应用程序域内。参考图 2，示出了一个示例性应用程序体系结构。聊天应用程序（ChatApplication）包括组件 ChatApplication.EXE 200 和 CAPICOMM.DLL 202。

通过采用共享名字空间模式（见 2000 年 6 月 28 日提交的名为“共享名字（Shared Names）”的申请号 09/605,602，其整体内容结合于此）可衍生出组件的唯一标识符—组件身份。组件身份包括组件名、版本和公钥令牌。类似地可以衍生应用程序身份。应用程序身份是通过应用程序或展开的相关图的路径（即，定制一个或多个应用程序的组件在应用程序或该组件或其应用程序所包含的其它组件的周围建立一名字解析范围），被表示为沿该路径的组件的组件身份的排序列表。该

路径上的最后一个组件通常是应用程序。

可以为聊天应用程序示例定义一清单。清单是包含关于组件的元数据的设计文档。考虑可以在用于任一组件的给定系统上有效的（组件绑定）政策状态，有效的清单是应用程序或展开的清单内容的编译，包括所有其成分和相关组件的清单的内容。另一方面，不考虑任一组件绑定政策并且不考虑任一外部/先决依赖（即，在 OS 组件上），合并应用程序清单是应用程序或展开的清单内容的编译，包括所有其构成组件的清单的内容。

再次参考图 2 的示例性应用程序体系结构，聊天应用程序具有将 ChatApplication.EXE 列出为构成文件的清单，包括该文件的散列。该清单也列出 CAPICOMM.DLL 为必须被安装/有效以使 ChatApplication.EXE 能够运行的依赖性。有效清单包含 ChatApplication.EXE 和 CAPICOMM.DLL 及其文件散列等等。合并应用程序清单仅包括聊天应用程序的清单的内容。由此，如果 CAPICOMM.DLL 被服务，则有效清单改变，但是合并应用程序清单不改变。应用程序身份是 ChatApplication.EXE 组件的身份（通过 ChatApplication.EXE 的相关图的普通单个节点路径），并且一与合并应用程序清单一样一即使 CAPICOMM.DLL 被服务也不改变。

在一个较佳实施例中，可以采用一基于声明性可扩充标记语言（XML）的模式来实现该应用程序清单。示例性实现如下：

```
<?xml version="1.0" encoding="utf-8"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
   manifestVersion="1.0" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2">

   <!-- Identify the application. -->
   <assemblyIdentity name="ChatApplication"
      version="1.0.0.0" publicKeyToken="0123456789abcdef"
      processorArchitecture="x86" language="neutral"/>

   <description asmv2:iconFile="chat.ico" />

   <!-- Identify the configuration file. -->
   <asmv2:configuration configFile="char.exe.config"/>

   <!-- Identify the application security requirements. -->
   <asmv2:TrustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
      <Security>
```

```
            <ApplicationRequestMinimum>
               <PermissionSet ID="A">
                  <IPermission class="IsolatedStorageFilePermission"
                     version="1" Allowed="DomainIsolationByUser"
                     UserQuota="1024000"/>
                  <IPermission class="UIPermission" version="1"
                     Window="SafeTopLevelWindows"
Clipboard="AllClipboard"/>
               </PermissionSet>
               <PermissionSet ID="FT">
                  <IPermission class="UIPermission" version="1"
                     Window="SafeTopLevelWindows"
Clipboard="AllClipboard"/>
               </PermissionSet>
               <AssemblyRequest name="MyDll_1"
PermissionSetReference="FT"/>
               <AssemblyRequest name="MyDll_2" PermissionSetReference="A"/>
            </ApplicationRequestMinimum>
         </Security>
      </asmv2:TrustInfo>

      <!-- Identify the main code entrypoint. -->
      <!-- This code runs the main method in an executable assembly. -->
      <entryPoint name="main" xmlns="urn:schemas-microsoft-com:asm.v2"
         dependencyName="MainAppAssembly">
      </entryPoint>

      <!-- Identify assembly dependencies. -->
      <!-- This code identifies a ChatApplication assembly. -->
      <dependency asmv2:name="MainAppAssembly">
         <dependentAssembly>
            <assemblyIdentity name="ChatApplication" version="1.0.0.0"
               publicKeyToken="e8ed396099c4b4e9"
processorArchitecture="x86"
               language="neutral"/>
         </dependentAssembly>
         <asmv2:installFrom codebase="ChatApplication.exe" size="1234"
               hash="EA016BCDD422E82D0A6E9FFA41771364" hashalg="SHA1"/>
```

```
    </dependency>

    <!-- This code identifies a CapiComm assembly. -->
    <dependency asmv2:name="CapiComm">
       <dependentAssembly>
          <assemblyIdentity name="CapiComm" version="1.0.0.0"
             publicKeyToken="e8ed396099c4b4e9"
processorArchitecture="x86"
             language="neutral"/>
       </dependentAssembly>
       <asmv2:installFrom codebase="Capicomm.dll" size="1234"
             hash="EA016BCDD422E82D0A6E9FFA41771364" hashalg="SHA1"/>
    </dependency>

<!-- This code identifies a CapiComm resource or satellite assembly. -->
       <dependency resourceType="resources" resourceFallbackCulture="en-
       us" resourceFallbackCultureInternal="false">
            <dependentAssembly>
               <assemblyIdentity name="CapiComm.Resources"
       version="1.0.0.0" publicKeyToken="0123456789abcdef" culture="*" />
            </dependentAssembly>
</dependency>

    <!-- Identify non-assembly files. -->
    <file name="chat.ico" hash="A78A91FF7A8C59192EDC05466A68BEE5"
       hashalg="SHA1" asmv2:size="12345"/>
    <file name="ChatApplication.exe.config"
hash="24C0C9E616CE459B730BD50128F361DE"
       hashalg="SHA1" asmv2:size="12345"/>
    <file name="eula.txt" hash="EA016BCDD422E82D0A6E9FFA41771364"
       hashalg="SHA1" asmv2:size="12345"/>
</assembly>
```

如上所述，许多应用程序展现允许应用程序实质上被定制或甚至加商标的设定。在许多情形下，甚至可能必须允许同一应用程序的多个定制同时在同一机器上安装。本发明在克服与采用本领域现有技术获得这一结果相关的困难中尤其有用。转向图 3，示出了一个示例性展开情形。在该情形中，聊天应用程序当前由两个站点/展开者定制：MSNBC 302 和 MSN 300。为启用这些情形，应用程序身份携带代码身份之外的附加信息；通过一单独的清单（“展开清单”）向定制（“展开”）分配其自己的身份（展开身份）。可以扩充以上示图以用连续的展开清单向该图添

加更多的定制。

在一个较佳实施例中，可以采用一基于声明性 XML 的模式来实现该展开清单。一个示例性实现如下：

```
<!-- Identify the deployment. -->
<assemblyIdentity name="MSNChatDeployment"
   version="1.0.0.0" publicKeyToken="0123456789abcdef"
   processorArchitecture="x86" language="neutral"/>

<!-- Specify application attributes. -->
<description xmlns="urn:schemas-microsoft-com:asm.v2"
   publisher="Microsoft" product="MSNChat"
   supportUrl="http://www.microsoft.com/deployments.asp?
               support=Microsoft.MSN.Chat.Deployment"/>

<!-- Specify the deployment attributes. -->
<deployment xmlns="urn:schemas-microsoft-com:asm.v2" >
   <!-- Create shell shortcuts and an Add or Remove Progams item. -->
   <install shellVisible="false"/>
</deployment>
```

```
    <!-- Identify the assembly dependencies -->
    <!-- This code specifies the base application manifest -->
    <dependency>
       <dependentAssembly>
            <assemblyIdentity name="ChatApplication"
            version="1.0.0.0" publicKeyToken="0123456789abcdef"
            processorArchitecture="x86" language="neutral"/>
       </dependentAssembly>
       <asmv2:installFrom

codebase="1.0.0.0/neutral/Microsoft.ChatApplication.manifest"/>
    </dependency>

   <dependency resourceType="resources">
        <dependentAssembly optional="yes">
          <assemblyIdentity name="CapiComm.Resources" version="1.0.0.0"
   publicKeyToken="0123456789abcdef" culture="fr" />
        </dependentAssembly>
</dependency>

    <file name="ChatApplication.exe.config"
hash="24C0C9E616CE459B730BD50128F361DF"
      hashalg="SHA1" asmv2:size="12350"/>
```

以上示例示出了 MSN 聊天 300 是如何能够为聊天应用程序定制安装和 UI 体验，使其被加上商标且看上去像 MSN。它也提示了 MSN 聊天 300 如何能够通过首要配置文件配置该应用程序。例如，如果配置文件具有设定背景色＝蓝（BackgroundColor=Blue），则由该展开提供的配置文件可推翻这一设定，令背景色＝白（BackgroundColor=White）。另外，该示例也示出了如何添加语言辅助（Language satellite），诸如如上所述用于 CapiComm 的法语辅助。

最后，该系统可使用证书和其它权限管理技术来确保展开者仅以可接受/应用程序作者授权的方式定制应用程序。在上述示例中，所示出的一种普通（trivial）方式是应用程序和展开者都具有同一"公钥"，由汇编身份中的公钥令牌属性证明。这确保了它们具有可信任的关系。

当用户装入快捷方式，该快捷方式将包含完整的应用程序身份（AppID）。这意味着如果定制该应用程序，该应用程序身份包括展开以及该应用的全身份。例如：

Converter.deploy, Version=1.0.0.0, Culture=x-ww,

PublicKeyToken=0123456789abcdef, ProcessorArchitecture=x86/Converter,

Version=1.0.0.0, Culture=x-ww, PublicKeyToken=0123456789abcdef,

ProcessorArchitecture=x86

为解析这一身份，绑定器遍历该身份的每一部分，并确定组件是否同步（即，是否与服务器语义同步），并且如果有中央服务政策，解析为该组件的适当的版本。这一过程确保了对每一个别的组件的服务。然后使用其运行时（runtime）实例中表示全身份的身份装入该应用程序。这通常被称为 AppID。AppID 可以在运行时由实体使用，如状态基础结构、信任管理器等等，以允许定制的应用程序的可配置性和可管理性。使用 AppID 的潜在操作包括：

·激活：给定应用程序身份，装入应用程序；

·安全：关于应用程序的政策状态，使用应用身份（参考）以将政策状态的作用域定到一个或多个应用程序；

·标识：从运行的应用程序的环境内确定应用程序身份；以及

·管理：确定安装在系统上的应用程序—应用程序管理 API 返回应用程序身份（定义）；卸载采用应用程序身份（定义），等等。

绑定器其本身也可以使用 AppID 以及与该 ID 的每一部分关联的清单组（AppID 可以用于获取“有效清单”或“合并应用程序清单”，并由此用于该结构内的任一数据），以提供对来自该清单的汇编或其它 DLL、及其位置信息的正确版本的绑定的支持。绑定器具有十分选择性地为定制的应用程序选择正确的绑定环境的能力。这允许系统提供一种模型，其中一个展开（或定制的应用程序）可使用组件的(多个)版本而不干涉同一应用程序的其它定制版本或完全不同的应用程序。这一情形依赖于基础组件可隔离的能力。

鉴于可应用本发明的原理的许多可能的实施例，应当认识到，此处参考附图所描述的实施例仅为说明性的，不应当限制本发明的范围。例如，鉴于性能原因，本发明的框架可以以硬件而不是软件实现。因此，此处描述的本发明设想处于所附权利要求书及其等效技术方案的范围之内的所有这样的实施例。

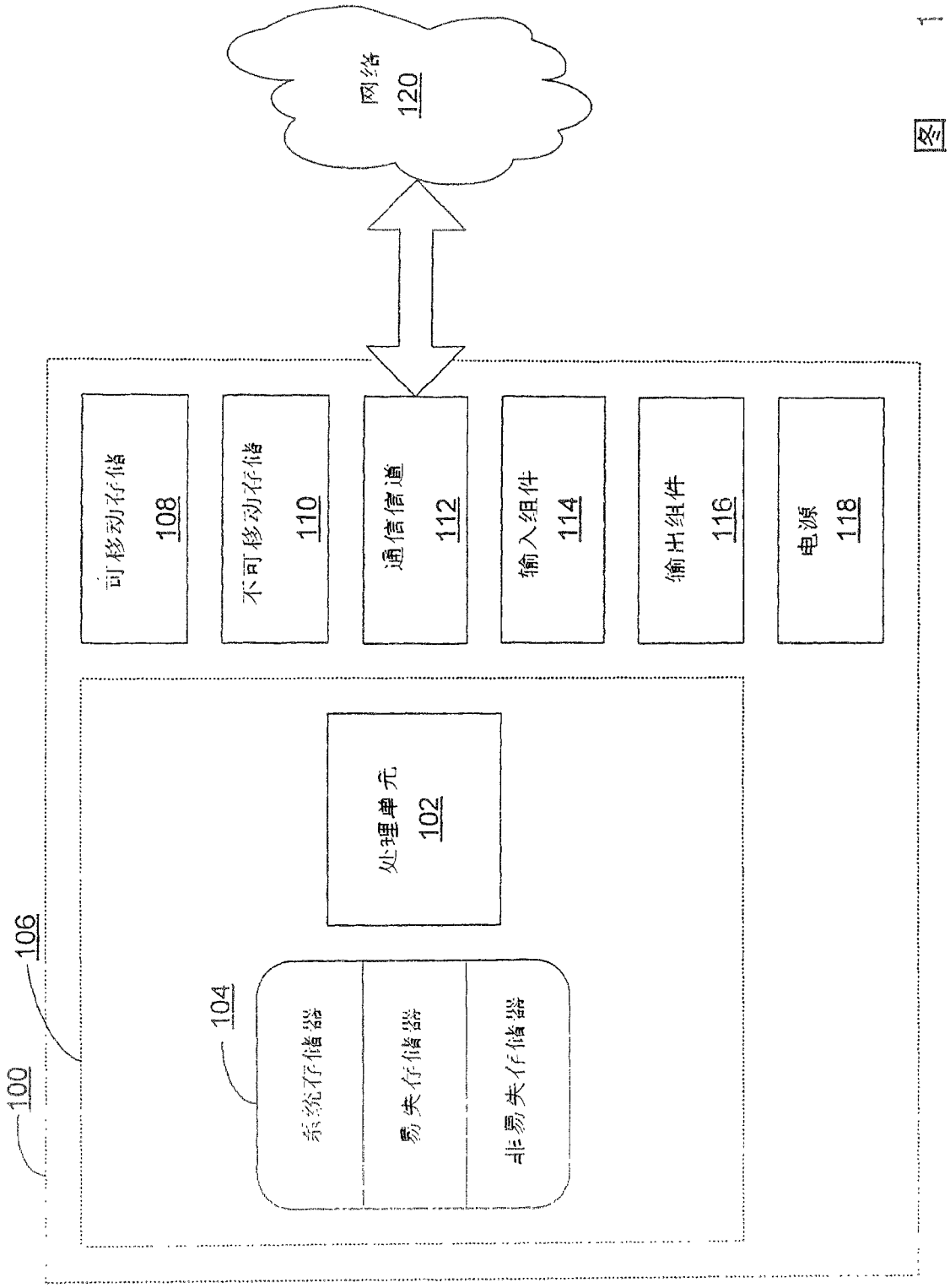

网络
120
可移动存储
108
不可移动存储
110
通信信道
112
输入组件
114
输出组件
116
电源
118
处理单元
102
系统存储器
易失存储器
非易失存储器
104
106
100

图 1

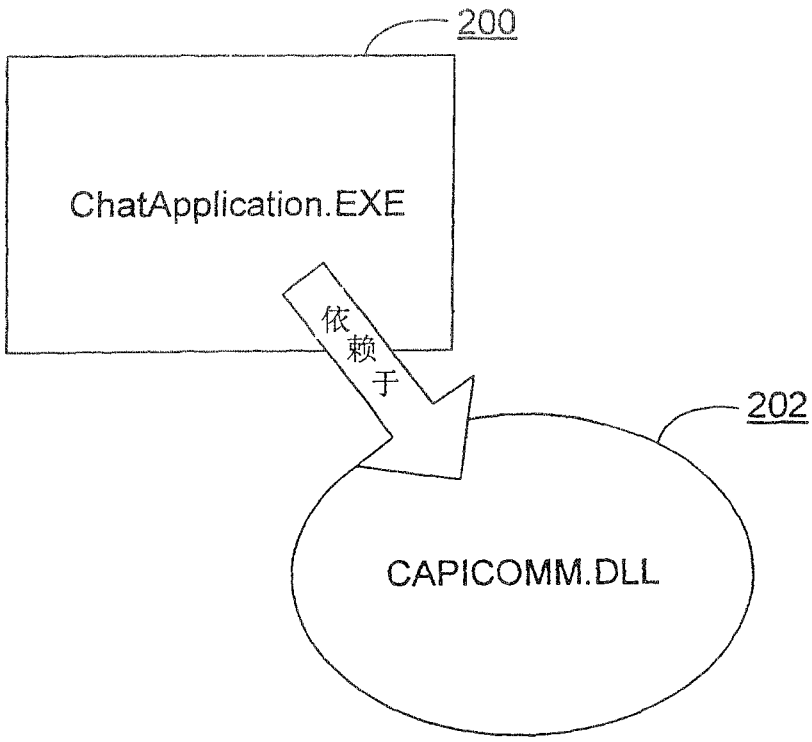


图 2

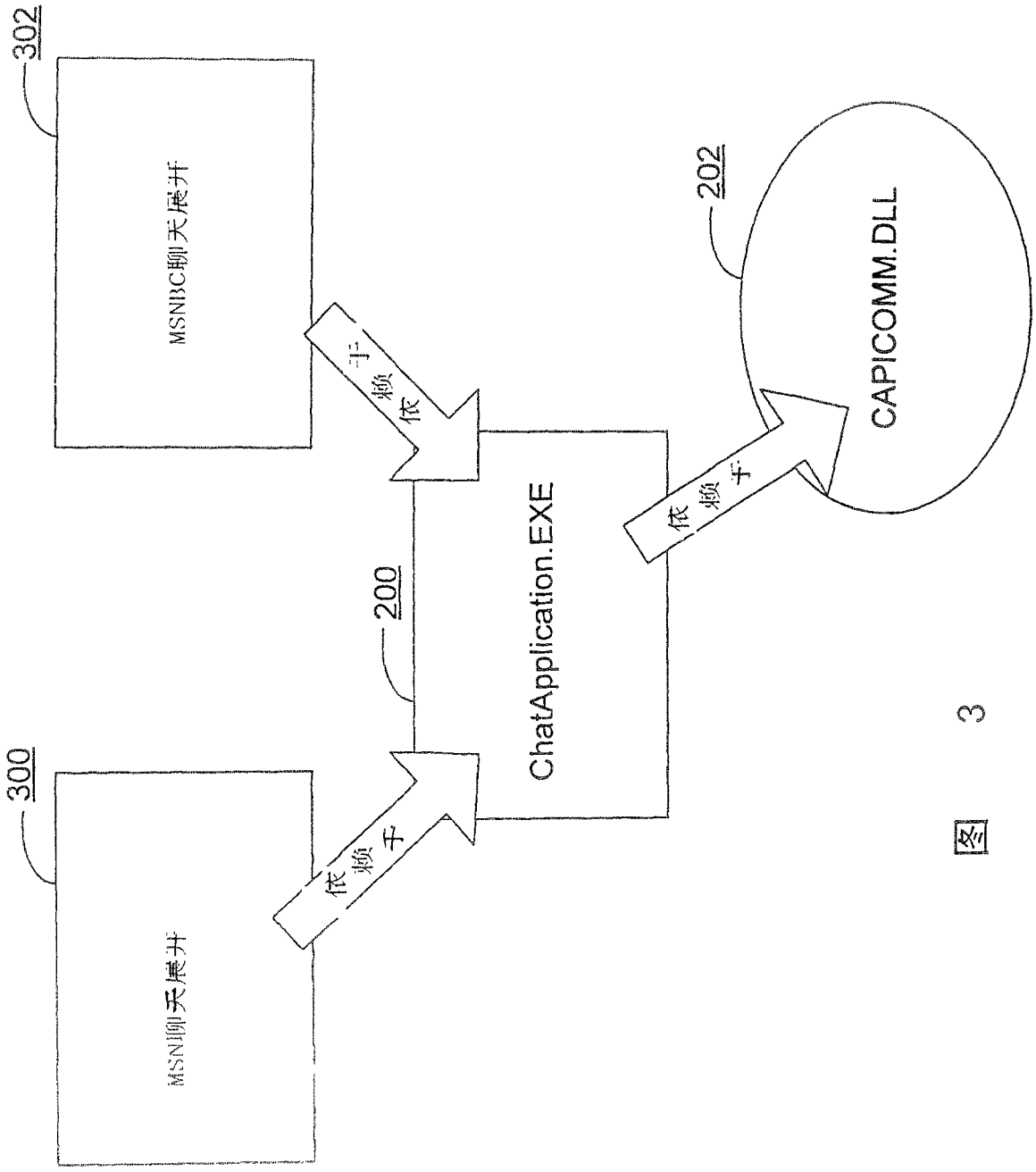

302
MSNBC聊天展开
300
MSN聊天展开
200
ChatApplication.EXE
202
CAPICOMM.DLL
依赖于
依赖于
依赖于

图 3