[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)
H04L 29/06 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200510125057.5

[43] 公开日 2006 年 6 月 7 日

[11] 公开号 CN 1783019A

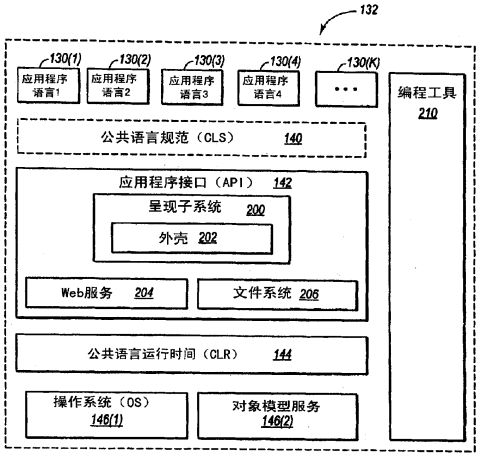
[22] 申请日 2005.11.18
[21] 申请号 200510125057.5
[30] 优先权
[32] 2004.12.3 [33] US [31] 11/003,307
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 A·A·斯特恩 A·M·德雅纳特
A·M·李四世 A·卡斯欧拉斯
C·D·尤帕德亚 C·G·卡勒
C·A·科利奇勒 D·沃腾迪科
D·E·莱维恩 D·O·德拉弗
D·A·沃尔特 D·M·普迪
E·L·维恩古尔德
E·B·克里斯滕森
E·P·亨尼科特
E·欧索弗特司基 G·L·埃普雷
G·M·德拉-利贝拉
J·E·约翰逊
J·瑞兹-斯考高尔 J·D·多蒂
J·T·惠勒 K·古普塔
K·D·沃尔夫 S·斯利尼瓦萨恩
L·E·欧尔森 M·T·塔维斯
M·奥塔维亚尼 M·A·范戈尔德
M·J·库尔森 M·J·马鲁切克
M·S·威尔诺 M·T·戴斯
M·马卡瑞奇安
N·H·杰塔南达尼 R·D·希尔
R·迪耶文多夫 R·T·斯特戈尔
S·诺格 S·C·希利 S·科恩
S·J·米尔雷特
S·T·斯瓦特兹 T·维森瓦纳斯
T·扬克苏科 U·S·赫格德
U·马丹 V·K·盖贾拉
V·A·莫迪 Y·佩萨奇
Y·肖豪德 V·B·巴莱约干
S·H·菲瑞斯 S·巴特瑞斯
S·普瑞 S·斯韦德罗弗
M·-H·E·拉马丹
K·斯利尼瓦萨恩 E·什维茨
A·拉雅戈帕兰
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 张政权

[54] 发明名称

用于创建 WEB 服务并与其交互的接口基础结构

[57] 摘要

Web 服务名字空间涉及用于实现各种应用程序的创建的基础结构。该基础结构提供用于构建各种规模和复杂程度的基于消息的应用程序的基础。该基础结构或框架提供用于基本消息通信、安全消息通信、可靠消息通信和事务化消息通信的 API。在一些实施例中，相关联的 API 以平衡效用、可使用性、可扩展行和可版本化性的方式被分解到名字空间的分层结构中。



权利要求书 4 页 说明书 25 页 附图 9 页

1．一个或多个计算机可读介质，其上存储有：

第一多个 web 服务 API，其每一个都用于编写分布式消息传递应用程序，其中，所述第一多个 web 服务 API 中的每一个都由第一公共名字空间前缀表示，用于将所述多个 web 服务逻辑分组；以及

第二多个 web 服务 API，其每一个都用于编写分布式消息传递应用程序并具有某些共同特征，所述共同特征并非在所有所述第一多个 web 服务 API 中都存在，其中，所述第二多个 web 服务 API 中的每一个都被表示为既有所述第一公共名字空间前缀又有第二公共名字空间前缀，所述第二公共名字空间前缀是所述第一公共名字空间前缀的子名字空间。

2．如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，所述第一多个 web 服务 API 被表示为源代码。

3．如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，所述第一多个 web 服务 API 被表示为已编译的代码。

4．如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，所述第一和第二多个 web 服务 API 可在.NET 平台上操作。

5．如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，所述第一公共名字空间是“System.ServiceModel”，或从文本“System.ServiceModel”编译得到。

6．如权利要求 5 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel.Channels”，或从文本“System.ServiceModel.Channels”编译得到。

7．如权利要求 5 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel.Configuration”，或从文本“System.ServiceModel.Configuration”编译得到。

8．如权利要求 5 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel.Design”，或从文本“System.ServiceModel.Design”编译得到。

9．如权利要求 5 所述的一个或多个计算机可读介质，其特征在于，所述第二

公共名字空间是“System.ServiceModel.Diagnostics”，或从文本“System.ServiceModel.Diagnostics”编译得到。

10. 如权利要求 5 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel.Security”，或从文本“System.ServiceModel.Security”编译得到。

11. 如权利要求 10 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel.Security.Protocols”，或从文本“System.ServiceModel.Security.Protocols”编译得到。

12. 如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，其上还具有：

第三多个 web 服务 API，其每一个都用于编写分布式消息传递应用程序并具有某些共同特征，所述共同特征并非在所有所述第一多个 web 服务 API 中都存在，其中，所述第三多个 web 服务 API 中的每一个都被表示为既有所述第一公共名字空间前缀又有第三公共名字空间前缀，所述第三公共名字空间前缀是所述第一公共名字空间前缀的子名字空间。

13. 如权利要求 12 所述的一个或多个计算机可读介质，其特征在于，其上还具有：

第四多个 web 服务 API，其每一个都用于编写分布式消息传递应用程序并具有某些共同特征，所述共同特征并非在所有所述第三多个 web 服务 API 中都存在，其中，所述第四多个 web 服务 API 中的每一个都被表示为既有所述第一公共名字空间前缀和所述第三公共名字空间前缀，又有第四公共名字空间前缀，所述第四公共名字空间前缀是所述第三公共名字空间前缀的子名字空间。

14. 如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，其上还具有：

第三多个 web 服务 API，其每一个都用于编写分布式消息传递应用程序并具有某些共同特征，所述共同特征并非在所有所述第一多个 web 服务 API 中都存在，其中所述第三多个 web 服务 API 中的每一个都被表示为既有所述第一公共名字空间前缀和所述第二名字空间前缀，又有第三公共名字空间前缀，所述第三公共名字空间前缀是所述第二公共名字空间前缀的子名字空间。

15. 如权利要求 1 所述的一个或多个计算机可读介质，其特征在于，所述一个或多个计算机可读介质是物理介质。

16．一个或多个计算机可读介质，其上具有：

第一多个 API，其每一个都用于实现具有第一一个或多个共同特征的功能，其中，所述第一多个 API 中的每一个都用第一公共名字空间前缀表示，用于将所述第一多个 API 逻辑分组；以及

第二多个 API，其每一个都用于实现具有第二一个或多个共同特征的功能，其中，所述第二多个 API 中的每一个都被表示为既有所述第一公共名字空间前缀，又有第二公共名字空间前缀，所述第二公共名字空间前缀是所述第一公共名字空间前缀的子名字空间。

17．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第一多个 API 被表示为源代码。

18．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第一多个 API 被表示为已编译的代码。

19．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第一和第二多个 API 可在.NET 平台上操作。

20．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.Security.Authorization”，或从文本“System.Security.Authorization”编译得到。

21．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.Runtime.Serialization”，或从文本“System.Runtime.Serialization”编译得到。

22．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.Transactions”，或从文本“System.Transactions”编译得到。

23．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.IO.Log”，或从文本“System.IO.Log”编译得到。

24．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.Text”，或从文本“System.Text”编译得到。

25．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.Xml”，或从文本“System.Xml”编译得到。

26．如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述第二公共名字空间是“System.ServiceModel”，或从文本“System.ServiceModel”编

译得到。

27. 如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述消息传递应用程序实现 Web 服务。

28. 如权利要求 16 所述的一个或多个计算机可读介质，其特征在于，所述一个或多个计算机可读介质是物理介质。

用于创建 WEB 服务并与其交互的接口基础结构

技术领域

本发明涉及软件开发。特别地，本发明涉及便于由应用程序和计算机硬件使用软件平台来创建分布式计算活动并在其中交互的应用程序接口（API）。

背景技术

计算技术已经改变了我们工作和娱乐的方式。计算系统如今有各种各样的形式，包括台式计算机、膝上计算机、图形输入板 PC、个人数字助理（PDA）、家用设备、等等。在其最基本的形式中，计算系统包括系统存储器以及一个或多个处理器。系统存储器中的软件可由处理器执行以指导该计算系统的其它硬件执行所需的功能。

软件一般被分成“操作系统”软件和“应用程序”软件。尽管计算系统并非绝对必须具有操作系统，但是操作系统在通用计算系统中是有帮助的，因为它们管理和控制计算硬件，并执行诸如文件管理、线程调度、多任务等通用系统任务。应用程序软件可执行更专用的任务，并且在必要时可调用操作系统所提供的系统级功能。操作系统的存在使应用程序开发变得更加流水线化，因为无须为每一个应用程序重新开发基本功能。

操作系统使许多内核功能对应用程序软件可用。应用程序开发者可通过草拟符合应用程序接口（API）的源代码来使应用程序软件调用操作系统所给出的功能。在运行时期间，当应用程序执行从该源代码编译所得的处理器级指令时调用操作系统。由此，应用程序软件通过调用单独的 API 功能来请求资源。API 功能还作为操作系统可向应用程序软件返回任何有关信息的手段。术语 API 用来指对操作系统的单个功能调用，以及对操作系统的可能的功能调用的集合。此外，术语 API 既适用于功能调用的源代码表示，又适用于功能调用的存储器内部表示。

在过去的几年里，因特网以及一般意义上的网络技术的普遍采用改变了计算机软件开发者的前景。以往，软件开发者集中于独立台式计算机、或经由局域网（LAN）连接到有限数量的其它计算机的基于 LAN 的计算机的单站点软件应用程

序。这些应用程序利用定义良好的 API 来访问计算机的底层操作系统。

随着因特网发展并获得更加广泛的接收，业界开始认识到在全球网（或简称“Web”）上的各个站点主存应用程序的功效。在联网的世界中，来自任何地方的客户机可提交对主存在各不同位置的基于服务器的应用程序的请求，并在几分之一秒内收回响应。但是，这些 Web 应用程序通常是使用相同的操作系统平台来开发的，而这些操作系统最初是为独立计算机器或本地联网的计算机开发的。不幸的是，在某些实例中，这些应用程序不能恰当转换成分布式计算模式。底层的平台不是简单地以支持无限个数的互连计算机的概念来构造的。

为了适应由因特网所引领的向分布式计算环境的转变，微软公司开发出一种称为“.NET”框架（读作“点 Net”）或 Microsoft® .NET 的软件平台。Microsoft® .NET 是用于连接人、信息、系统和设备的软件。该平台允许开发者创建通过因特网执行的 Web 服务。此动态转变随附.NET 框架的 API 功能集合。

随着.NET 框架的使用变得越来越普遍，提高该平台效率和/或性能的方法已被标识出来。发明者已开发出允许此类提高的效率和/或性能的独特的 API 功能集合。

发明内容

本发明的原理针对在与使用 web 服务名字空间的.NET 框架交互时、允许提高的效率和性能的独特的 API 功能集合。Web 服务名字空间涉及用于允许创建各种应用程序的基础结构，而术语“web”并不试图限制或以任何方式约束使用 web 服务名字空间的应用程序仅被用于因特网应用程序。该基础结构提供用于构建各种规模和复杂程度的基于消息的应用程序的基础。该基础结构或框架提供用于基本消息通信、安全消息通信、可靠消息通信和事务化消息通信的 API。在一些实施例中，相关联的 API 以平衡效用、可使用性、可扩展性和可版本化性的方式被分解到名字空间的分层结构中。

本发明的其它特征和优点将在以下描述中阐述，一部分将从该描述中变得显而易见，或可通过实施本发明来学得。本发明的特征和优点可通过在所附权利要求书中特别指出的装置和组合的手段来实现并获得。本发明的这些及其它特征将从以下描述和所附权利要求书中变得更透彻，或可通过如以下所述地实施本发明来得悉。

附图说明

为了描述可获得本发明上述及其它优点和特征的方式，将参考在附图中示出的本发明的具体实施例来提供对以上所简述的本发明的更具体的描述，附图中相同的标号指相同的元素。理解了这些附图仅描绘了本发明的典型实施例且因而不应被示为限制其范围，将通过使用附图更具体地描述本发明，附图中：

图 1 示出客户机使用常规协议访问 Web 服务的网络体系结构；

图 2 是一种包括应用程序接口（API）的网络平台的软件体系结构的框图；

图 3 是 API 所支持的唯一名字空间的框图，其中每个名字空间表示具有共同特征的一个或多个功能类；

图 4 是可执行整个软件体系结构或其部分的示例性计算机的框图；

图 5 是根据本发明的一个实施例描述编程接口的若干方面的框图；

图 6 是根据一个实施例描述编程接口的若干方面的框图；

图 7 是根据一个实施例描述编程接口的若干方面的框图；

图 8 是根据一个实施例描述编程接口的若干方面的框图；

图 9 是根据一个实施例描述编程接口的若干方面的框图；

图 10 是根据一个实施例描述编程接口的若干方面的框图；

图 11 是根据一个实施例描述编程接口的若干方面的框图；

图 12 是根据一个实施例描述编程接口的若干方面的框图；

图 13 是根据一个实施例描述编程接口的若干方面的框图；

图 14 是根据一个实施例描述编程接口的若干方面的框图；

图 15 是根据一个实施例描述编程接口的若干方面的框图；以及

图 16 是根据一个实施例描述编程接口的若干方面的框图。

具体实施方式

本发明的原理涉及在与使用 web 服务名字空间的.NET 框架交互时、允许提高的效率和性能的独特的应用程序（API）功能集合。Web 服务名字空间涉及用于允许创建各种应用程序的基础结构，该基础结构提供用于构建各种规模和复杂程度的基于消息的应用程序的基础。该基础结构或框架提供用于基本消息通信、安全消息通信、可靠消息通信和事务化消息通信的 API。在一些实施例中，相关联的 API 以平衡效用、可使用性、可扩展性和可版本化性的方式被分解到名字空间的分层结构中。

在所述实现中，网络平台使用可扩展标记语言（XML），它是一种用于描述

数据的开放式标准。XML 由全球网协会（W3C）管理。XML 用于定义 Web 页面和商务到商务文档上的数据元素。XML 使用和 HTML 类似的标签结构；但是，HTML 定义如何显示元素，而 XML 定义那些元素包含什么。HTML 使用预定义标签，但 XML 允许文档的开发者定义标签。因此，实际上任何数据项都可被标识，从而允许 XML 文档像数据库记录那样工作。通过使用 XML 和诸如简单对象访问协议（SOAP）等其它开放式协议，该平台允许可根据用户需要定制的很大范围的服务的集成。尽管本文中所描述的实施例是结合 XML 和其它开放式标准描述的，但这些不是操作所要求保护的发明所必须的。其它同样可行的技术将足以实现本文中所描述的发明。

如本文中所使用，术语“应用程序接口”或 API 包括使用方法或功能调用的常规接口，以及使用远程调用（例如，代理、承接关系）和 SOAP/XML 调用的接口。

示例性网络环境

图 1 示出可实现诸如.NET 框架等网络平台的网络环境 100。网络环境 100 包括代表性的 Web 服务 102(1)、……、102(N)，它们提供可通过网络 104（例如，因特网）或使用进程间通信可被访问的服务。统常由标号 102 引用的 Web 服务是可重复使用并可通过网络 104 编程交互（通常是通过工业标准 Web 协议，诸如 XML、简单对象访问协议（SOAP）、无线应用程序协议（WAP）、超文本传输协议（HTTP）、以及简单邮件传输协议（SMTP），尽管也可使用诸如远程过程调用（RPC）或对象代理类型技术等通过网络与 Web 服务交互的其它手段的可编程应用程序组件。Web 服务可以是自描述的，并且常以消息的格式和排序的形式来定义。

Web 服务 102 可由其它服务（如由通信链路 106 所表示）或诸如 Web 应用程序 110（如由通信链路 112 和 114 所表示）等软件应用程序直接访问。每一个 Web 服务 102 都被示为包括执行软件以处理对特定服务的请求的一个或多个服务器。Web 服务可被配置成执行各种不同服务中的任何一种。Web 服务的例子包括登录验证、通知、数据库存储、库存报价、位置目录、映射关系、音乐、电子钱包、日历/调度程序、电话列表、新闻和信息、游戏、票务、等等。各种 Web 服务可相互结合或与其它应用程序结合，以构建可变复杂系统和智能交互式体验。

网络环境 100 还包括利用 Web 服务 102（如通信链路 122 所示）和/或 Web 应用程序 110（如通信链路 124、126 和 128 所示）的代表性客户机设备 120(1)、120(2)、120(3)、120(4)、……、120(M)。客户机也可使用标准协议相互通信，如

客户机 120(3)和 120(4)之间的示例性 XML 链路 130 所示。

一般由标号 120 引用的客户机设备能以许多不同方式实现。可能的客户机实现的例子包括，但不限于，便携式计算机、固定计算机、图形输入板 PC、电视/机顶盒、无线通信设备、个人数字助理、游戏控制台、打印机、复印机、以及其它智能设备。

Web 服务应用程序 110 是设计成在网络平台上运行、并可在处理和服务来自客户机 120 的请求时利用 Web 服务 102 的应用程序。Web 服务应用程序 110 由在编程框架 132 上运行的一个或多个软件应用程序 130 组成，这些软件应用程序在一个或多个服务器 134 或其它计算机系统上执行。注意，Web 服务应用程序 110 的一部分实际上可驻留在一个或多个计算机 120 上。或者，Web 服务应用程序 110 可与客户机 120 或其它计算机上的其它软件协调以完成其任务。

编程框架 132 是支持应用程序开发者所开发的应用程序和服务的结构。它通过支持多编程语言来允许多语言开发和无缝集成。它支持诸如 SOAP 等开放式协议，并封装底层操作系统和对象模型服务。该框架为多编程语言提供稳健和安全的执行环境，并提供安全、集成的类库。

框架 132 是多层体系结构，包括 API 层 142、公共语言运行时间库（CLR）层 144、以及操作系统/服务层 146。此分层体系结构允许对各层的更新和修改，而不会影响框架的其它部分。公共语言规范（CLS）140 允许各种语言的设计者编写能够访问底层库功能的代码。规范 140 起到语言设计者和库设计者之间的契约的作用，它可被用于促进语言的可相互操作性。通过遵守 CLS，用一种语言编写的库可由用另一种语言编写的代码模块直接访问，以实现用一种语言编写的代码模块和用另一种语言编写的代码模块之间的无缝集成。CLS 的一个示例性详细实现在由 ECMA TC39/TG3 的参与者所创建的 ECMA 标准中描述。

API 层 142 给出可由应用程序 130 调用以访问层 146 所提供的资源和服务的若干组功能。通过给出网络平台的 API 功能，应用程序开发者可为充分利用本地和网络资源以及其它 Web 服务的分布式计算系统创建 Web 服务应用程序，而无需理解那些网络资源如何实际操作或如果使那些网络资源可供使用的复杂的相互作用。此外，Web 服务应用程序可以用任何数量的编程语言编写，并被翻译成公共语言运行时间 144 所支持的中间语言，并作为公共语言规范 140 的一部分被包括。以此方式，API 层 142 可为各种各样的应用程序提供方法。

此外，框架 132 可被配置成支持从主宿该框架的服务器 134 以外的远程设备

上执行的远程应用程序所进行的 API 调用。分别驻留在客户机 120(3)和 120(M)上的代表性应用程序 148(1)和 148(2)可通过网络 104 直接或间接调用 API 层 142 来使用 API 功能。

该框架还可在客户机处实现。客户机 120(3)表示框架 150 在客户机处实现的情况。此框架可与基于服务器的框架 132 完全相同，或为客户机目的进行修改。或者，在客户机是诸如蜂窝电话、个人数字助理、手持式计算机或其它通信/计算设备等有限或专用功能设备的情况下，基于客户机的框架可被压缩精简。

开发者的编程框架

图 2 更详细地示出编程框架 132。公共语言规范（CLS）层 140 支持用各种语言 130(1)、130(2)、130(3)、130(4)、……、130(K)编写的应用程序。此类应用程序语言包括 Visual Basic、C++、C#、COBOL、Jscript、Perl、Eiffel、Python、等等。公共语言规范 140 指定特征或关于特征的规则的一子集，当遵守这些特征或特征规则，则允许各种语言进行通信。例如，某些语言不支持给定类型（例如，“int*”类型），而公共语言运行库 144 可能反而支持该类型。在此情形中，公共语言规范 140 不包括该类型。另一方面，所有或大多数语言所支持的类型（例如，“int[]”类型）被包括在公共语言规范 140 中，从而库开发可自由使用该类型，并可确保该语言能够处理该类型。

此通信能力导致用一种语言编写的代码模块和用另一种语言编写的代码模块之间的无缝集成。因为不同的语言特别适合于特定的任务，所以语言之间的无缝集成允许开发者为特定代码模块选择特定语言，同时能够随用不同语言编写的模块使用该代码模块。公共语言运行库 144 允许具有跨语言继承的无缝多语言开发，并为多编程语言提供稳健和安全的执行环境。公共语言规范 140 和公共语言运行时间 144 的更多信息，读者请参看 2000 年 6 月 21 日提交的题为“Method and System for Compiling Multiple Languages”（用于编译多语言的方法和系统）（序列号 09/598,105)和 2000 年 7 月 10 日提交的题为“Unified Data Type System and Method”（统一数据类型系统和方法）（序列号 09/613,289）的共同待批美国专利申请，其内容通过引用包含与此。

框架 132 封装操作系统 146(1)（例如，Windows®操作系统）和对象模型服务 146(2)（例如，组件对象模型（COM）或分布式 COM）。操作系统 146(1)提供常规功能，诸如文件管理、通知、事件处理、用户界面（例如，窗口、菜单、对话框、等等）、安全、认证、验证、进程和线程、存储器管理、等等。对象模型服务 146(2)

提供与其它对象的接口以执行各种任务。对 API 层 142 的调用被传递给公共语言运行时间层 144，以由操作系统 146(1)和/或对象模型服务 146(2)本地执行。

API 142 将 API 功能组成多个名字空间。名字空间本质上定义了提供有关功能的特定集合的、统称为“类型”的类、接口、委托、枚举和结构的集合。类表示被分配了数据的受管制堆，它具有引用赋值语义。委过是面向对象的类型安全的函数指针。枚举是一类特殊的值类型，它可取若干由被命名的常量表示的预定值中的一个。结构表示具有赋值语义的静态的被分配数据。接口定义其它类型可实现的契约。

通过使用名字空间，设计者可将类型集合组织成分层结构的名字空间。设计者能够从类型集合创建多个组，其中每一组都至少包含一个类型，该类型给出逻辑相关的功能。在示例性实现中，API 142 被组织成包括 3 个根名字空间。应当注意，尽管仅在图 2 中示出 3 个根名字空间，但是还可在 API 142 中包括额外的根名字空间。API 142 中所示的 3 个名字空间是：呈现子系统（包括用户界面外壳的名字空间 202）的第一名字空间 200、web 服务的第二名字空间 204、以及文件系统的第三名字空间 206。每个组随即被赋予一个名字。例如，呈现子系统名字空间 200 中的类型可被赋予名字“Windows”（窗口），而文件系统名字空间 206 中的类型可被赋予名字“Storage”（存储）。已命名的组可被组织到系统级 API 的单个“全局根”名字空间下，诸如总系统名字空间等。

通过选择顶层标识符并将其作为前缀，每个组中的类型可很容易地由分层结构名引用，分成结构名包括包含该类型的组的名字加到所选择的顶层标识符作为前缀。例如，文件系统名字空间 206 中的类型可使用分层结构名“System.Storage”（系统.存储）来引用。以此方式，个体名字空间 200、204 和 206 成为系统名字空间下的主分支，并可携带个体名字空间加上诸如“System.”（系统.）前缀等指示符作为前缀的名称。

呈现子系统名字空间 200 涉及编程和内容开发。它提供允许应用程序、文档、媒体呈现及其它内容的生成的类型。例如，呈现子系统名字空间 200 提供一编程模型，它允许开发从操作系统 146(1)和/或对象模型服务 146(2)获得服务。外壳名字空间 202 涉及用户界面功能。它提供允许开发者将用户界面功能嵌入其应用程序、并进一步允许开发者扩展用户界面功能的类型。

Web 服务名字空间 204 涉及用于允许各种应用程序（例如，简单如在内联网的两个同辈之间操作的聊天应用程序，和/或复杂如供百万用户使用的可调节 Web 服务的应用程序）的创建的基础结构。所描述的基础结构有利的是高度可变，因为

仅需使用适宜于特定解决方案的复杂程度的那些部分。该基础结构提供用于构造各种规模和复杂程度的基于消息的应用程序的基础。框架的基础结构为 API 提供基本消息通信、安全消息通信、可靠消息通信和事务化消息通信。在下述实施例中，相关联的 API 以精心制作成平衡效用、可使用性、可扩展性和可版本化性的方式，被分解到名字空间的分层结构中。

文件系统名字空间 206 涉及存储。它提供允许信息存储和检索的类型。

除了框架 132 以外，还提供编程工具 210 来帮助开发者构建 Web 服务和/或应用程序。编程工具 210 的一个例子是 Visual Studio®，它是微软公司提供的一种多语言编程工具套件。

根名字空间

图 3 更详细地示出 API 142 和其根名字空间中的一个，即，web 服务 204。在一个实施例中，名字空间是根据一种分层结构的命名惯例来标识的，在该分层结构的命名惯例中，名字串是用句号来串接的。虑及此命名惯例，以下提供所选择的 API 142 的名字空间的概览，尽管可等效地使用其它命名惯例。

Web 服务名字空间 204 由根名字“System.ServiceModel”（系统.服务模型）标识。在“System.ServiceModel”名字空间内是以下子名字空间：

- 通道（即，“System. ServiceModel.Channels”），
- 配置（即，“System.ServiceModel.Configuration”），
- 设计（即，“System.ServiceModel.Design”），
- 诊断（即，“System.ServiceModel.Diagnostics”），以及
- 安全（即，“System.ServiceModel.Security”）。

此外，在“System.ServiceModel.Security”名字空间内有以下子名字空间：

- 协议（即，“System.ServiceModel.Security.Protocols”）。

此外，有支持除消息通信以外的和具有消息通信以外的效用的功能的其它名字空间。这些名字空间包括“System.Security.Authorization”（系统.安全.授权）、“System.Runtime.Serialization”（系统.运行库.串行化）、“System.Transactions”（系统.事务）、“System.IO.Log”（系统.IO.日志）、“System.Text”（系统.文本）以及“System.XML”（系统.XML）。

现在将更详细地描述所示名字空间中的每一个及其对应的内核类。

System.ServiceModel

System.ServiceModel 是封装所有服务模型的根名字空间。具体而言，此名字

空间表示用来编写使用 Web 服务消息的分布式、消息传递应用程序的 API。此 API 集合被分组到一起，因为它表示服务模型的逻辑的内核层，并且能够仅使用那些 API 而不使用其它任何服务模型名字空间来编写应用程序。此名字空间的内核类包括以下：

- BasicProfileHttpBinding（基本概况 Http 绑定）- 内建的 WS-I BP 1.1 绑定。
- BasicProfileHttpsBinding（基本概要 Https 绑定）- HTTPS 上内建的 WS-I BP 1.1 绑定。
- ChannelFactory<T>（通道工厂<T>）- 用于创建客户机方的运行时间。
- ContractDescription（契约描述）- 描述服务契约。包含 OperationDescription（操作描述）的集合，其中每一个都包含 MessageDescription（消息描述）的集合。
- EndpointAddress（端点地址）- EndpointAddress（及包含该类的 AddressProperties（地址属性））表示 Web 服务的地址的定义。它包括涉及服务地址的若干方面，包括其 SOAP 地址（ActingAs）、其传输地址（Address）、元数据贡献的头部（EndpointHeaders）、元数据不可知的头部（InstanceHeader）、安全身份（Identity）、以及关于在该端点处实现的绑定和契约的信息。
- IChannelFactory（I 通道工厂）- IChannelFactory 接口表示用于主动创建通道的基本接口。其上有两个方法集合 – CanCreateChannel<T>()，（能创建通道<T>()），它允许用户查询是否支持特定类型的 IChannel；以及 CreateChannel<T>()，（创建通道<T>()），它允许用户创建到指定端点特定类型的 IChannel。
- IListenerFactory（I 监听器工厂）- IListenerFactory 接口表示用于在一网络地址处监听并创建 IListener 的基本接口。它给出功能的类型 – 配置网络地址的方法（ListenUri、SetListenUri、SetUniqueListenUri）（监听 Uri、设置监听 Uri、设置唯一监听 Uri）以及创建 IListener 的方法（CreateListener<T>(...)、CanCreateListener<T>()）（创建监听器<T>(...)、能创建监听器<T>()）。

IListener 一般是通过传入消息过滤器在 IListenerFactories 上创建的。在监听器工厂接收到的以及匹配监听器的过滤器的消息被传递给该监听器。这可与 TCP/IP 工作的方式类比——计算机（监听器工厂）在特定 IP 地址上监听。套接字绑定到特定 TCP 端口（过滤器）。特定 IP 地址上到达的分组基于端口号被多路分解成若干套接字，类似于到达特定监听器工厂的消息基于过滤器被多路分解成若干监听器。IListenerFactories 一般由通信模式的接受者使用。通道如何被活动创建详见 IChannelFactory。

• IntermediaryHttpBinding – 中间 HTTP 绑定。

• IntermediaryTcpBinding – 中间 TCP 绑定。

• Message（消息）– 消息是服务模型中各应用程序之间通信的基本单元。它是封装 Web 服务之间所有数据交换的容器。它是等效于 IP 分组的服务模型。消息的结构松散地对应于 SOAP 包封。消息既包含消息头部集合又包含正文，它们分别对应于 SOAP 头部块和 SOAP 正文。可包括在消息中的头部集合是可扩展的。若干默认头部类型被定义和指定，具体而言这些涉及消息定址和消息到期的头部类型。其它头部类型由其它服务模型组件和/或由第三方开发者定义。

• MessageDescription（消息描述）- 描述服务消息。

• MsmqsBinding（Msmqs 绑定）– MSMQ 集成绑定。

• NetMsmqBinding（网络 Msmq 绑定）- MSMQ 绑定。

• NetMsmqsBinding（网络 Msmqs 绑定）- 安全 MSMQ 绑定。

• NetNamedPipeBinding（网络已命名管道绑定）- 已命名管道绑定。

• NetTcpBinding（网络 Tcp 绑定）- 不可相互操作的 TCP 绑定。

• NetTcpsBinding（网络 Tcps 绑定）- 不可相互操作的 TCPS 绑定。

• OperationAttribute（操作属性）- 当根据.NET 框架属性模型使用时，此属性将一方法标记为服务契约的一部分，并允许开发者配置该操作的各个方面。

• OperationDescription（操作描述）- 描述服务操作。包含 MessageDescription 的集合。

• SecurityBindingElement（安全绑定元素）- 此类及其相关类（SOAPAuthenticationMode、SecurityContextMode、ChannelSecurityCredentials、ServiceSecurityCredentials、ChannelSecurityBehaviors、ServiceSecurityBehaviors）（SOAP 认证模式、安全上下文模式、通道安全凭证、服务安全凭证、通道安全行为、服务安全行为）提供用于在通道堆栈中配置 SOAP 安全的框架。即，它们确定要被应用于消息的安全类型。

• ServiceContractAttribute（服务契约属性）- 当根据.NET 框架属性模型使用时，此属性将一接口标记为服务契约，并允许开发者配置该契约的各个方面。

• ServiceDescription（服务描述）- 服务的运行时间要求的存储器内部表示，这些要求可用于各种目的，包括构建合适的运行时环境，生成元数据，以及编码或配置信息。

• ServiceEndpoint（服务端点）- 表示与世界通信的入口，并包含端点地址、

契约和绑定。

• ServiceHost<T>（服务主机<T>）- 随通信能力提供具体服务类型的运行时间容器。

• WsHttpBinding（WsHttp 绑定）- HTTP 上可相互操作的 WS（非 BP）绑定。

• WsHttpsBinding（WsHttps 绑定）- HTTPS 上可相互操作的 WS（非 BP）绑定。

System.ServiceModel.Channels

System.ServiceModel.Channel 是服务模型传输、可靠性以及通道工厂和监听器工厂排队的的主名字空间。这些类表示 Web 服务通信子系统的具体实现。此 API 集合被分组到一起，因为它表示用于对 SOAP 消息进行作用的逻辑、整体的通信 API 层。此名字空间的内核类包括以下：

• BinaryMessageEncoder（二进制消息编码器）- BinaryMessageEncoder 表示使用消息的.NET-二进制格式编码将消息对象转换成一系列八位字节及其逆操作的抽象。它定义取流或数组片段<字节>并返回消息对象实例、以及取消息并将该消息写流到或数组片段<字节>中的方法。

• HttpChannelFactory（Http 通道工厂）- HttpChannelFactory 表示用于活动创建 HTTP 通道的基类。HTTP 通道用于使用 SOAP-over-HTTP 协议向服务发送消息，并可能从服务接收响应。

• HttpListenerFactory（Http 监听器工厂）- HttpLIstenerFactory 类表示用于在 HTTP 地址监听 SOAP 消息并接受启动为该地址的通道的基类。经由 SOAP-over-HTTP 协议通信的消息可在通道上接受，或在 HttpListenerFactory（及其帮助程序 IListener）下接受。

• NamedPipeChannelFactory（已命名的管道通道工厂）- NamedPipeListenerFactory 类表示用于在已命名的管道地址处监听 SOAP 消息并接受启动为该地址的通道的基类。经由.NET-消息成帧协议通过 Windows 已命名管道通信的消息可在通道上接受，或在 NamedPipeListenerFactory（及其帮助程序 IListener）下接受。

• ReliableChannelFactory（可靠通道工厂）- ReliableChannelFactory 类表示用于主动创建可靠通道的基类。可靠通道用于使用 WS 可靠消息通信协议通过任何受支持的传输向服务发送消息，并可能从服务接受响应。

• ReliableListenerFactory（可靠监听器工厂）- 该可靠类表示用于监听新的可

靠通道的基类。使用 WS 可靠消息通信协议通信的消息可在通道上接受，或在 ReliableListenerFactory（及其帮助程序 IListener）下接受。

• TcpChannelFactory（Tcp 通道工厂）- TcpChannelFactory 类表示用于活动创建 TCP 通道的基类。TCP 通道用于使用.NET 消息成帧协议通过 TCP 流向服务发送消息，并可能从服务接受响应。

• TcpListenerFactory(Tcp 监听器工厂)- TcpListenerFactory 类表示用于在 TCP 地址处监听 SOAP 消息并接受启动为该地址的通道的基类。经由.NET 消息成帧协议通过 TCP 流通信的消息可在通道上接受，或在 TcpListenerFactory（及其帮助程序 IListener）下接受。

• TextMessageEncoder（文本消息编码器）- TextMessageEncoder 表示使用标准的 XML 消息编码将消息对象转换为一系列八位字节及其逆操作的抽象。它定义取流或数组片段<字节>并返回实例化消息、以及取消息并将该消息写到流或数组片段<字节>中的方法。

System.ServiceModel.Configuration

System.ServiceModel.Configuration 名字空间包含用于存储服务模型应用程序的持久设置的类。这些 API 被分组在一起，因为它们在应用程序的开发后首先被访问。这些类并非由开发者在编写服务模型应用程序时使用，而是在应用程序被编写完以后使用。该单独的名字空间明确此区别。此名字空间的内核类包括以下：

• ServiceBindingSection（服务绑定区）- ServiceBindingSection 包含特定服务模型绑定的所有配置。这些配置设置的功能严密地反映出该编程模型中等价类的功能。

• ServiceModelSectionGroup（服务模型区组）- ServiceModelSectionGroup 是所有服务模型配置的容器类。它还是在编程读或写所有其它服务模型配置时使用的访问者类。

• ServicesSection（服务区）- ServiceSection 包含在.NET 框架应用程序域中实现服务的所有配置设置。

System.ServiceModel.Design

System.ServiceModel.Design 名字空间包含处理变换描述（ServiceDescription 和 ChannelDescription）的类型。这些包括从代码和配置创建描述的 ServiceLoader（服务加载器）和 ChannelLoader（通道加载器），以及导入和导出元数据和代码生成器的 ServiceDescriptionImporter（服务描述导入器）和

ServiceDescriptionExporter（服务描述导出器）。它还包括从 ServiceDescription 和 ChannelDescription 构建运行时间环境的类型。这些类型被分组到一个名字空间下，因为它们全部以某种方式变换 ServiceDescription 和 ChannelDescription。该名字空间以 Design 结尾，因为其所支持的许多情形是严格的设计时间情形或任何消息交换以前的运行时间情形（可将其视为“自建立的应用程序”情形）。此名字空间的内核类包括以下：

• ChannelBuilder(通道构建器)- 基于 ServiceDescription 和 ChannelDescription 构建监听器和通道堆栈。

• ChannelLoader（通道加载器）- 基于已定类型的通道和相关联的绑定构建 ChannelDescription。

• ServiceDescriptionImporter/Exporter（服务描述导入器/导出器）- 从元数据生成 ServiceDescription 及其逆操作。

• ServiceLoader（服务加载器）- 从服务类型以及相关联的契约和绑定创建 ServiceDescriptio。

System.ServiceModel.Diagnostics

System.ServiceModel.Diagnostics 名字空间给出用于通过对运行时间状态的直接检查和控制、以及通过持久跟踪流来监视和诊断服务模型应用程序的类型。使用这些类的情形设计观察和控制服务模型应用程序的行为，而不是创建应用程序。这些 API 被分组成该名字空间以明确该功能区别。此名字空间的内核类包括以下：

• IWSTransferContract（IWS 传输契约）- IWSTransferContract 是实现 WS-传输协议的接口。WS-传输是 WS-管理协议套件中的一部分，并被用于检查 Web 服务应用程序的运行实例。

• MessageWriterTraceListener（消息写入器跟踪监听器）- MessageWriterTraceListener 是 System.Diagnostics.TraceListener（系统.诊断.跟踪监听器）类的实现，它用于记录 Web 服务的传入和传出消息。

System.ServiceModel.Security

System.ServiceModel.Security 名字空间包含涉及消息安全的类型。它包含 SOAP 消息安全、WS-信任、WS-安全对话和其它 WS-*安全协议的实现。它包含若干个类，这些类用于获取安全令牌、用于将安全令牌串行化、用于通过在 System.Security.Authorization 名字空间中调用类型来验证安全令牌、以及用于通过在 System.Security.Authorization 名字空间中调用类型来执行授权检查。它还包含安

全绑定抽象（安全绑定是一种消息安全模式）。此名字空间中的内核类包括以下：

• SecurityBindingFactory（安全绑定工厂）- SecurityBindingFactory 是管理安全模板（绑定）的框架。安全模板（绑定）是符合 SOAP 消息安全协议的安全消息的特定模式。它根据模板保护传出消息以及验证传入消息。此类创建 SecurityBinding（安全绑定）的实例，后者是派生所有安全绑定的抽象基类。SecurityBinding 类提供 SecureOutgoingMessage（保护传出消息）和 ValidateIncomingMessage（验证传入消息）的功能。

• SecurityBindingFactory 在 ServiceModel.Security.Protocols（服务模型.安全.协议）名字空间中具有特定实现。以下列出其中一部分：

- – AnonymousOverAsymmetricSecurityBindingFactory
  （非对称上匿名的安全绑定工厂）
- – BasicOverAsymmetricSecurityBindingFactory
  （非对称上基本的安全绑定工厂）
- – BasicOverSymmetricSecurityBindingFactory
  （对称上基本的安全绑定工厂）
- – BasicOverTransportSecurityBindingFactory
  （传输上基本的安全绑定工厂）
- – CryptoOverAsymmetricSecurityBindingFactory
  （非对称上加密的安全绑定工厂）
- – CryptoOverTransportSecurityBindingFactory
  （传输上加密的安全绑定工厂）
- – SessionSecurityBindingFactory
  （会话安全绑定工厂）
- – SoapSecurityBindingFactory
  （SOAP 安全绑定工厂）
- – SymmetricSecurityBindingFactory
  （对称安全绑定工厂）
- – TransportSecurityBindingFactory
  （传输安全绑定工厂）

• SecurityTokenAuthenticator（安全令牌认证器）- 令牌认证器存储应被用于令牌验证、并在给定其类型专属部分前提下将令牌实例化的认证设置。

• SecurityTokenProvider（安全令牌提供器）- SecurityTokenProvider 提供获得用于在远程端点处认证主参与者的安全令牌的功能。

• SecurityTokenResolver（安全令牌解析器）- SecurityToeknResolver 是创建和匹配对令牌的引用（以及发现先决令牌）的公共框架。

• SecurityTokenSerializer（安全令牌串行化器）- SecurityTokenSerializer 将 SecurityToken（安全令牌）串行化为 XML，将令牌 XML 去串行化为类型专属的格式中性的部分，并调用 TokenAuthenticator（令牌认证器）来生成 securityToken（安全令牌）。它还知道如何对内部和外部令牌引用进行串行化和去串行化。

System.ServiceModel.Security.Protocols

此名字空间包含匹配某些良好定义和分析的安全模式（称为安全绑定）的 SOAP 消息安全的具体实现。它包含 System.ServiceModel.Security 中定义的框架的具体实现。

System.Security.Authorization

System.Security.Authorization 名字空间包含负责涉及安全令牌、声明和授权的内核安全功能的所有的类。在服务模型名字空间以外单独的名字空间给出此功能，以允许其它消息通信框架利用其所给出的安全功能。此名字包括以下内核类：

• IClaim（I 声明）- 安全模型的基础是声明——即发布方所作的语句。声明由最低程度地实现 IClaim 接口的对象表示。IClaim 接口进而实现 IMatchPolicy（I 匹配策略）接口。此接口给出匹配方法。这用于确定指定的声明是否“匹配”实现该匹配方法的对象。从 IClaim 导出新的接口以创建声明分类法。最终由对象实现这些接口，而这些接口本身也可派生为其它接口。此名字空间定义 4 个顶层类型的声明：身份、属性、发布和访问决策。身份声明作出关于如何标识一方的语句。这可以是主要的或关键的持有器。属性声明是关于诸如电话号码或生日等特性的语句。发布声明指示所指定的身份被允许发布特定类型的声明。访问决策声明是已对所指示的身份授权或授予的权利或能力。

• IClaimSet（I 声明集合）- 在此模型中，声明只是单个语句。因此，接下来的逻辑组件是声明集合。这些是由实现 IClaimSet 接口的对象提供的。声明集合可具有任意的一组声明，唯一的例外是它仅能具有单个关键的承载声明。

• ICropto（I 加密）- ICrypto 接口标识关键持有器能够执行的加密操作。该加密操作既可以是对称的，也可以是非对称的。

• ISecurityToken（I 安全令牌）- 安全令牌是支持 IClaimsProvider（I 声明提

供器）和 ICryptoProvider（I 加密提供器）接口的对象，这意味着它是声明集合的容器，并给出加密操作的加密提供器。验证方法被调用以检查和验证令牌。验证确保所有发布的权威机构签名都被确认，并有效地将那些权威机构的声明集合与其自己的声明集合合并，从而发布委托链可作为授权的一部分被验证。

System.Runtime.Serialization

System.Runtime.Serialization 名字空间包含 XML 格式化器类（服务模型的主串行化引擎）、有关的类、以及用于标记可串行化的类的定制属性和接口——这些属性和接口构成服务模型的串行化编程模型。此名字包括以下内核类：

• DataContractAttribute（数据契约属性）- 此定制属性用于声明一类型是可串行化的，并已根据新的服务模型串行化编程模型注释，并为该类型设置某些串行化属性。

• DataMemberAttribute(数据成员属性)- 在新的服务模型串行化编程模型中，此定制属性用于声明给定字段或属性应被串行化。它还可用于控制该域或属性的某些串行化设置。

• KnownTypeAttribute（已知类型属性）- 此定制属性用于向“已知类型集合”添加类型，“已知类型集合”是串行化引擎在其不知道要将对象去串行化成哪种类型的情况下将会尝试的类型集合。

• UnknownSerializationData（未知串行化数据）- 此类用于同一对象的不同版本之间信息的往返：当将对象较新版本的实例去串行化为较旧的版本时，较旧版本所不理解的任何信息都存储在 UnknownSerializationData 中。当较旧的对象被串行化时，存储在 UnknownSerializationData 中的数据也被串行化，以确保在已串行化的数据然后要被去串行化回较新的版本的情况下没有数据丢失。

• XmlFormatter（Xml 格式化器）- 此类是服务模型的主串行化引擎——通过调用此类上的方法，可启动串行化或去串行化操作。

System.Transactions

System.Transactions 名字空间包括两个子名字空间：System.Transactions.Isolation（系统.事务.隔离）和 System.Transaction.Recovery（系统.事务.恢复）。

System.Transactions.Isolation

System.Transactions.Isolation 名字空间给予应用程序在协同操作的客户机之间隔离资源的能力。该名字空间中的类型支持易失性和可持续性可锁定资源的隔离，

以及可变粒度资源支持。此名字空间包括以下内核类：

- LockContext（锁定上下文）- 表示是代表其锁定资源的客户机上下文。
- LockManager（锁定管理器）- 锁的集合表示，提供锁的生命期管理。
- LockScope（锁定范围）- 表示对资源的操作的边界。
- ResourceLock（资源锁）- 表示锁。

System.Transactions.Recovery

System.Transactions.Recovery 名字空间提供帮助资源管理器的开发者的日志记录对象模型。日志类支持不同的工作单位、补偿记录和检查点。此名字空间包括以下内核类：

- Log（日志）- 表示资源管理器的恢复日志。
- LogRecord（日志记录）- 表示写入恢复日志的单个记录
- SavePoint - 表示一个单位的工作中的保存点，并提供截取、合并和回卷日志的方法。
- UnitOfWork（工作单位）- 表示在资源管理器的日志中的工作的逻辑单位，并提供用于准备、提交和回卷该工作单位的方法。

System.IO.Log

System.IO.Log 名字空间包含提供到面向记录的串行化 I/O 系统的简单接口的类。它还包含到 WINDOWS® CLFS（公共日志文件系统）的受管制接口。此名字空间包括以下内核类：

- IRecordSequence（I 记录序列）- 提供记录序列抽象的接口。

System.Text

System.Text 名字空间容纳服务模型 XML 基础结构组件（Xml 读出器和写入器）用于执行各种形式的二进制/文本编码和解码（Base64 和 BinHex）的帮助程序类。此名字空间包括以下内核类：

- Base64Encoding（Base64 编码）- 以 Base64 格式处理二进制数据的编码和解码。
- BinHexEncoding（BinHex 编码）- 以 BinHex 格式处理二进制数据的编码和解码。

System.XML

System.Xml 包含服务模型“XML 基础结构”——服务模型串行化所使用的优化的 XML 读出器和写入器，但也可独立使用。当前所支持的读出器和写入器的两

个集合是UTF8文本XML的性能已优化的读出器和写入器，以及服务模型专用的二进制格式的读出器/写入器。此名字空间包括以下内核类：

• IXmlDictionary（IXml字典）- 实现IXmlDictionary的类可担当XmlDictionaryString（XML字典串）对象的资料档案库。

• XmlBinaryReader（Xml二进制读出器）、XmlBinaryWriter（Xml二进制写入器）- 这些类以服务模型的专用二进制格式读和写XML。

• XmlDictionaryReader（Xml字典读出器）、XmlDictionaryWriter（Xml字典写入器）- 这些抽象类构成此名字空间内所有其它读出器和写入器的基础，并对标准XML读出器和写入器API引入允许使用串字典（见XmlDictionaryString）的新方法。即，可用XmlDictionaryString代替普通串来调用WriteString（写入串），且如果它支持写入器，则可仅写入串的字典ID（而不是整个串），从而实现空间节约。

• XmlDictionaryString（Xml字典串）- 此类本质上用于将串与唯一的ID号结合，从而如果该串被多次写出（使用支持此特征的XML写入器），则该串可仅被写出一次，然后在后续使用中由其唯一ID所代替，以实现空间节约。

• XmlUTF8TextReader（XmlUTF8文本读出器）、XmlUTF8TextWriter（XmlUTF8文本写入器）- 这些类使用UTF8编码来读和写文本格式的XML。它们类似于标准文本XML读出器和写入器，但它们是经过性能优化的，并且是从使得它们在服务模型串行化引擎的已优化代码路径中可使用的XmlDictionaryReader/Writer导出的。

示例性计算系统和环境

图4示出可在其中实现（整体或部分地）编程框架132的合适的计算环境400的示例。计算环境400可在本文所描述的计算机和网络体系结构中使用。

示例性计算环境400仅仅是计算环境的一个例子，并不试图对计算机和网络体系结构的使用范围或功能提出任何限制。也不应将计算环境400解释为具有涉及在示例性计算环境400中示出的各组件中的任何一个或其组合具有任何依赖性或要求。

框架132可用许多其它通用或专用计算系统环境或配置来实现。可能适用的公知的计算系统、环境、和/或配置的例子包括，但不限于，个人计算机、服务器计算机、多处理器系统、基于微处理器的系统、网络PC、小型计算机、大型计算机、包括以上任何系统或设备的分布式计算环境、等等。还可在诸如蜂窝电话、个

人数字助理、手持式计算机、或其它通信/计算设备等客户机或有限资源中实现该框架的紧缩或子集版本。

框架 132 可在由一个或多个计算机或其它设备执行的诸如程序模块等计算机可执行指令的通用上下文中描述。一般而言，程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、数据结构、等等。框架 132 还可在分布式计算环境中实施，其中任务是由通过通信网络连接的远程处理设备执行的。在分布式计算环境中，程序模块可位于包括记忆存储设备的本地和远程计算机存储介质中。

计算环境 400 包括计算机 402 形式的通用计算设备。计算机 402 的组件可包括，但不限于，一个或多个处理器或处理单元 404、系统存储器 406、以及将包括处理器 404 在内的各种系统组件耦合到系统存储器 406 的系统总线 408。

系统总线 408 表示各种可能类型的总线结构中的一种或多种，包括存储器总线或存储器控制器、外围总线、加速图形端口、以及使用各种总线体系结构中的任何一种的处理器或局部总线。作为示例，此类体系结构可包括工业标准体系结构（ISA）总线、微通道体系结构（MCA）总线、增强 ISA（EISA）总线、视频电子标准协会（VESA）总线、以及也称为 Mezzanine 总线的外围组件互连（PCI）总线。

计算机 402 通常包括各种计算机可读介质。此类介质可以是计算机 402 可访问的任何可用介质，并包括易失性和非易失性、可移动和不可移动介质。

系统存储器 406 包括诸如随机存取存储器（RAM）410 等易失性存储器形式的计算机可读介质，和/或诸如只读存储器（ROM）412 等非易失性存储器。包含诸如在启动期间帮助在计算机 402 的各元件之间传输信息的基本例程的基本输入/输出系统（BIOS）414 存储在 ROM 412 中。RAM 410 通常包含可由处理单元 404 直接访问和/或正由其操作的数据和/或程序模块。

计算机 402 还可包括其它可移动/不可移动、易失性/非易失性计算机存储介质。作为示例，图 4 示出用于读和写不可移动、非易失性磁介质（未示出）的磁硬盘驱动器 416，用于读和写可移动、非易失性磁盘 420（例如，“软盘”）的磁盘驱动器 418，以及用于读和/或写诸如 CD-ROM、DVD-ROM 或其它光介质等可移动、非易失性光盘 424 的光盘驱动器 422。硬盘驱动器 416、磁盘驱动器 418 和光盘驱动器 422 每一个都由一个或多个数据介质接口 426 连接到系统总线 408。或者，硬盘驱动器 416、磁盘驱动器 418 和光盘驱动器 422 可由一个或多个接口（未示出）

连接到系统总线 408。

各盘驱动器及其相关联的计算机可读介质为计算机 402 提供计算机可读指令、数据结构、程序模块和其它数据的非易失性存储。尽管该例示出硬盘 416、可移动磁盘 420 和可移动光盘 424，但是应当理解，还可使用可存储可由计算机访问的数据的其它类型的计算机可读介质来实现该示例性计算系统和环境，诸如磁带盒或其它磁存储设备，闪存卡，CD-ROM、数字多功能盘（DVD）或其它光存储，随机存取存储器（RAM），只读存储器（ROM），电可擦除可编程只读存储器（EEPROM）等等。

可在硬盘 416、磁盘 420、光盘 424、ROM 412 和/或 RAM 410 上存储任意数量的程序模块，例如包括，操作系统 426、一个或多个应用程序 428、其它程序模块 430、以及程序数据 432。操作系统 426、一个或多个应用程序 428、其它程序模块 430、以及程序数据 432 中的每一个（或其某种组合）都可包括该编程框架 132 的元素。

用户可经由诸如键盘 434 和定位设备 436（例如，“鼠标”）等输入设备将命令和信息输入到计算机 402 中。其它输入设备 438（未具体示出）可包括话筒、操纵杆、游戏垫、圆盘式卫星天线、串行端口、扫描仪、等等。这些及其它输入设备经由耦合到系统总线 408 的输入/输出接口 440 连接到处理单元 404，但也可由诸如并行端口、游戏端口或通用串行总线（USB）等其它接口和总线结构连接。

监视器 442 或其它类型的显示设备也可经由诸如视频适配器 444 等接口连接到系统总线 408。除了监视器 442 以外，其它输出外围设备可包括可仅有输入/输出接口 440 连接到计算机 402 的诸如扬声器（未示出）和打印机 446 等组件。

计算机 402 可使用到诸如远程计算设备 448 等一个或多个远程计算机的逻辑连接在联网环境中工作。作为示例，远程计算设备 448 可以是个人计算机、便携式计算机、服务器、路由器、网络计算机、对等设备或其它普通网络节点、等等。图示远程计算设备 448 未可包括本文中相对于计算机 402 所描述的许多或全部元件和特征的便携式计算机。

计算机 402 和远程计算机 448 之间的逻辑连接被描述为局域网（LAN）450 和一般广域网（WAN）452。此类网络环境常见于办公室、企业范围的计算机网络、内联网和因特网。

当在 LAN 网络环境中实现时，计算机 402 经由网络接口或适配器 454 连接到局域网 450。当在 WAN 网络环境中实现时，计算机 402 通常包括调制解调器 456

或用于通过广域网 452 建立通信的其它装置。可内置或外置于计算机 402 的调制解调器 456 可仅有输入/输出接口 440 或其它适当机制连接到系统总线 408。应当认识到，所示网络连接时示例性的，并可使用在计算机 402 和 448 之间建立通信链路的其它装置。

在诸如以计算环境 400 所示等联网环境中，相对于计算机 402 所描述的各程序模块或其部分可存储在远程记忆存储设备中。作为示例，远程远程应用程序 458 驻留在远程计算机 448 的存储器设备上。为说明目的，应用程序和诸如操作系统等其它可执行程序组件在本文中被示为离散的框，尽管可认识到，此类程序和组件在各个时间驻留在计算设备 402 的不同存储组件中，并由计算机的数据处理器执行。

框架 132、特别是 API 142 或对 API 142 的调用的实现可存储在某种形式的计算机介质上，或通过其传输。计算机可读介质可以是可由计算机访问的任何可用介质。作为示例，而非限制，计算机可读介质可包括“计算机存储介质”和“通信介质”。“计算机存储介质”包括以用于存储诸如计算机可读指令、数据结构、程序模块或其它数据等信息的任何方法或技术实现的易失性和非易失性的、可移动和不可移动的介质。计算机存储介质包括，但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术，CD-ROM、数字多功能盘（DVD）或其它光存储，磁带盒、磁带、磁盘存储或其它磁存储设备，或可用于存储所需信息并可由计算机访问的任何其它介质。

“通信介质”通常具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块、或其它数据。通信介质还包括任何信息传递介质。术语“已调制数据信号”指已在信号中以将对信息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，而非限制，通信介质包括诸如有线网络或直接连线连接等有线介质，以及诸如声学、RF、红外、及其它无线介质等无线介质。以上的任意组合也包括在计算机可读介质的范围之内。

或者，该框架的部分可在硬件，或硬件、软件和/或固件的组合中实现。例如，一个或多个应用集成电路（ASIC）或可编程逻辑器件（PLD）可被设计或编程为实现该框架的一个或多个部分。

编程接口

编程接口可被视为用于使一个或多个代码片断能与一个或多个其它代码片断所提供的功能通信或访问这些功能的任何机制、过程、协议。一类编程接口是应用程序接口，它通常由应用程序调用。或者，编程接口可被视为能够通信地耦合到其

它组件的一个或多个机制、方法、功能调用、模块等的系统组件的一个或多个机制、方法、功能调用、模块、对象等。前句中的术语“代码片断”旨在包括一个或多个指令或代码行，并且包括例如，代码模块、对象、子例程、函数、等等，无论所应用的术语是什么，无论各代码片断是否是分开编译的，也无论各代码片断被提供为源、中间、还是对象代码，无论各代码片断是在运行时间系统还是过程中使用的，无论它们位于相同或不同的机器上还是分布在多个机器上，也无论各代码片断所标识的功能是完全用软件、完全用硬件、还是用软件和硬件的组合实现。

概念上，可在一般意义上看待编程接口，如图 5 或图 6 所示。图 5 将编程接口“接口 1”图示为一导管，第一和第二代码片断通过该导管进行通信。图 6 将编程接口图示为包括接口对象 11 和 12（它们可以是也可以不是第一和第二代码片断的部分），它们使系统的第一和第二代码片断能够仅通过介质 M 进行通信。参考图 6，可将编程接口对象 11 和 12 视为同一系统不同的编程接口，也可将对象 11 和 12 加上介质 M 视为组成编程接口。尽管图 5 和 6 示出双向流以及流的每一方上的接口，但是某些实现可能仅有一个方向的信息流（或如下所述没有信息流）或仅在一方具有接口对象。作为示例，而非限制，诸如应用程序编程接口（API）、入口点、方法、函数、子例程、远程过程调用和组件对象模型（COM）等术语都被包括在编程接口的定义内。

这样一个编程接口的各个方面可包括第一代码片断将信息（其中使用“信息”最宽泛的意义，并包括数据、命令、请求、等等）发送给第二代码片断的方法；第二代码片断接收该信息的方法；以及该信息的结构、序列、句法、组织、模式、定时和内容。在这点上，只要信息是以接口所定义的方式被传输，底层传输介质本身对于接口的操作就可能是无关紧要的，无论该介质是有线的、无线的、还是两者的组合。在某些情况下，信息可能不是以常规意义的一个或两个方向被传递，因为信息传递或可经由另一种机制（例如，放在缓冲区、文件等中的信息，独立于代码片断之间的信息流），或可为非现存的，如一代码片断简单地访问另一代码片断所执行的功能时的情形。任一或所有这些方面在给定情况下都可能是重要的，例如取决于代码片断是在松耦合还是紧耦合配置中的系统的一部分，因此该列表应被视为示例性的，而非限制性的。

此编程接口的概念对本领域技术人员而言是公知的，并且从本发明前述的详细描述是明确的。但是，还有实现编程接口的其它方法，并且除非明确排除在外，否则这些也是旨在为所附权利要求书所包括的。此类其它方法可能表现得比图 5

和 6 的简化视图更为精密或复杂，但是它们执行类似的功能以实现相同的整体效果。现在我们将简要描述编程接口的某些示例性替换实现。

分解

从一代码片断到另一代码片断的通信可通过将通信分成多个离散通信来间接实现。这在图 7 和 8 种示出。如图所示，能以可分功能集合的形式来描述一些接口。因此，图 5 和 6 的接口功能可被分解以达到相同效果，正如可数学地提供 24，或者 2 乘以 2 乘以 3 乘以 2。由此，如图 7 中所示，编程接口“接口 1”所提供的功能可被细分，以将该编程接口的通信转换为多个接口“接口 1A”、“接口 1B”、“接口 1C”等等，同时可达到相同效果。如图 8 所示，“接口 I1”所提供的功能可被细分为多个编程接口 I1a、I1b、I1c 等等，同时可达到相同效果。类似地，从第一代码片断接收信息的第二代码片断的编程接口 I2 可被分解成多个编程接口 I2a、I2b、I2c 等等。当进行分解时，第一代码片断中所包括的接口个数无需匹配第二代码片断中包括的编程接口个数。在图 7 和 8 的任何一例中，接口“接口 1”和 I1 的功能特质分别保持与图 5 和 6 相同。编程接口的分解还可遵循结合律、交换律、以及其它数学属性，以使分割很难被辨认。例如，操作的排序可能无关紧要，因此，编程接口所执行的功能可早在到达该接口之前由另一段或另一接口执行，或由系统的一个单独的组件执行。此外，编程领域普通技术人员可认识到，有各种调用不同功能达到相同结果的方法。

重新定义

在某些情形中，也许能够忽略、添加或重新定义编程接口的某些方面（例如，参数），同时仍然达到预期效果。这在图 9 和 10 中示出。例如，假设图 5 中的编程接口“接口 1”包括函数调用 Square(input, precision, output)，（平方(输入,精度,输出)）即包括 3 个参数，input、precision 和 output 的调用，并且是从第一代码片段向第二代码片段发布。如果在给定情形中，中间的参数 precision 无关紧要，如图 9 所示，则它也可被忽略，或甚至用无意义的（在此情况中）参数代替。还可添加其它无关紧要的参数。无论在哪种情况下，只要在第二代码片段将输入平方以后返回了输出，square 的功能即可被实现。Precision 对计算系统的一些下游或其它部分而言很可能是有意义的参数；但是，一旦认识到对于计算平方的狭隘目的而言，精度不是必须的，则其即可被替换或忽略。例如，可传递诸如生日等无意义的值来取代传递有效的精度值，从而不会对结果产生负面影响。类似地，如图 10 所示，接口 I1 被编程接口 I1’所取代，重新定义为忽略或向该接口添加参数。编程接口 I2

可被类似地重新定义为编程接口 I2'，重新定义为忽略非必要参数，或可在其它地方处理的参数。这里的要点是，在某些情形中，编程接口可包括诸如参数等对于某些目的而言非必要的方面，因此它们可被忽略或重新定义，或为其它目的在其它地方进行处理。

内联编码

将两个单独的代码模块的部分或所有功能合并，以使它们之间的“接口”改变形式也是可行的。例如，图 5 和 6 的功能分别可被转换为图 11 和 12 的功能。图 11 中，图 5 的第一和第二代码片段被合并成包含两者的一个模块。在此实例中，两个代码片段仍可相互通信，但接口可被改编为更适合单个模块的形式。因此，例如，可能不再需要正式的调用和返回语句，但依照编程接口“接口 1”的类似处理或响应可能仍然有效。类似地，如图 12 所示，图 6 的接口 I2 的部分或全部可被内联地写入编程接口 I1 中以构成编程接口 I1”。如图所示，编程接口 I2 被分成 I2a 和 I2b，且接口部分 I2a 已与编程接口 I1 内联编码来构成编程接口 I1”。举个具体例子，考虑图 6 的编程接口 I1 执行函数调用 square(input, output)，编程接口 I2 接收到该调用，在由第二代码片段处理了用 input 传递的值（求平方）以后，编程接口 I2 用 output 传回平方后的结果。在此实例中，由第二代码片段所执行的处理（将输入平方）可由第一代码片段执行，而无须调用该编程接口。

分离

从一个代码片段到另一个代码片段的通信可通过将该通信分成多个离散通信来间接完成。这在图 13 和 14 中示出。如图 13 所示，提供一件或多件中间件（分离接口，因为它们将功能和/或接口函数从原始接口分离）来转换第一接口“接口 1”上的通信，以使它们符合不同的接口，在此实例中为编程接口“接口 2A”、“接口 2B”和“接口 2C”。这在例如有已安装的、设计成根据接口 1 协议与例如操作系统通信的应用程序基础的地方可以实现，但是此后操作系统改为使用不同的接口，在此实例中为编程接口“接口 2A”、“接口 2B”和“接口 2C”。要点是，第二代码片段所使用的原始接口被改变，从而它不再与第一代码片段所使用的接口兼容，因此使用中介来使旧的和新的接口兼容。类似地，如图 14 所示，可引入第三代码片段，它用分离接口 DI1 从接口 I1 接收通信，并用分离接口 DI2 向例如重新设计成与 DI2 合作的接口 I2a 和 I2b 等发送接口功能，但该第三片段提供相同的功能效果。类似地，DI1 和 DI2 可协同工作以将图 6 的接口 I1 和 I2 的功能转换到新的操作系统，同时提供相同或类似的功能效果。

重写又一种可能的变体是动态地重写代码，用其它达到相同总体效果的功能来代替编程接口功能。例如，可有一种系统，其中在执行环境中（诸如“.NET”框架、Java 运行时间环境、或其它类似的运行时间类型的环境所提供的执行环境等），以中间语言（例如，Microsoft IL。Java ByteCode 等等）给出的代码片段被提供给即时（JIT）编译器或解释器。JIT 编译器可被写入以将通信从第一代码片段动态转换到第二代码片段，即，使通信符合不同的接口，如第二代码片段（无论是原始的还是不同的第二代码片段）所要求的。这在图 15 和 16 中示出。如图 15 中可看到，此方法类似于上述的分离情形。这在例如已安装的应用程序基础被设计成根据接口 1 协议与操作系统通信的的地方可以实现，但是此后操作系统改为使用不同的接口。JIT 编译器可被用于在运行中从已安装的基础应用程序的通信符合操作系统的新接口。如图 16 所示，此动态重写接口的方法可被应用于动态因素，或者也可用于改变接口。

还要注意，上述经由各替换实施例以编程接口形式达到相同或类似效果的各种情形还能以各种方式串行和/或并行组合，或与其它插入代码结合。因此，以上所给出的各替换实施例不是互斥的，而是可被混和、匹配和结合，以产生与图 5 和 6 中给出的一般情形相同或等效的情形。还要注意，如大多数编程构造那样，还有实现相同或相似接口功能的其它类似方法，这些方法未在本文中描述，但却由本发明的精神和范围所表示，即，应当注意，接口的价值至少部分地在于接口所表示的功能、以及接口所实现的有利效果。

结论

尽管用结构化特征和/或方法性动作专属的语言描述了本发明，但是应当理解，所附权利书中定义的本发明不必限于所述具体特征或动作。相反，揭示这些具体特征和动作是作为实现所要求保护的发明的示例性形式。

本发明可具体化为其它特定形式，而不会偏离其精神或本质特征。从任何意义上来说，所述实施例仅是示例性的，而不是限制性的。因此，本发明的范围由所附权利要求书指示，而不是由前述描述指示。落入所附权利要求书的等效技术方案的意义和范围之内的所有改变将被包括在其范围之内。

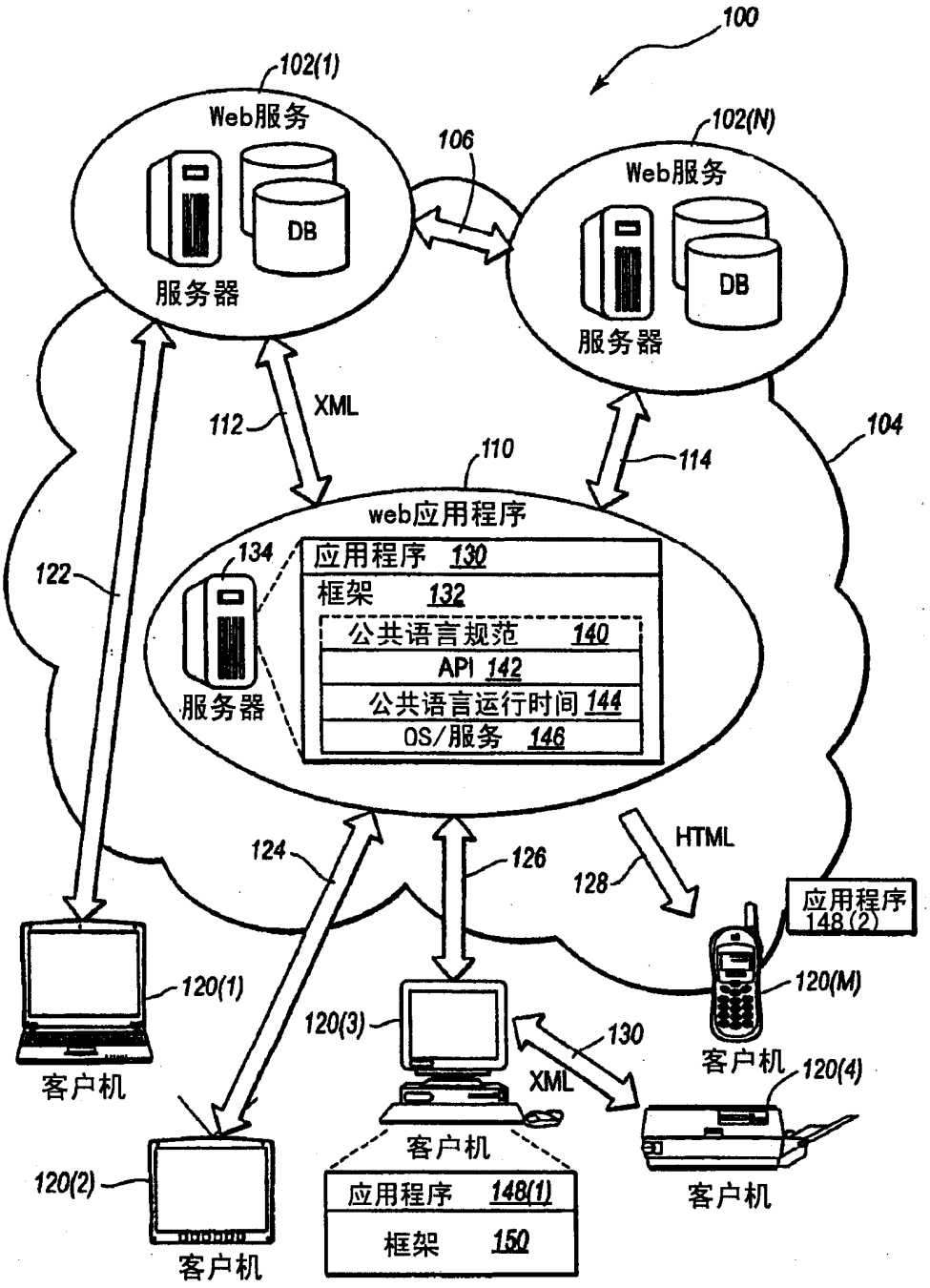

100
102(1)
Web服务
DB
服务器
106
102(N)
Web服务
DB
服务器
112
XML
110
114
104
web应用程序
134
应用程序 130
框架 132
公共语言规范 140
API 142
公共语言运行时间 144
OS/服务 146
服务器
122
124
126
HTML
128
应用程序 148(2)
120(1)
客户机
120(3)
130
XML
客户机
120(M)
客户机
120(4)
客户机
120(2)
客户机
应用程序 148(1)
框架 150

图 1

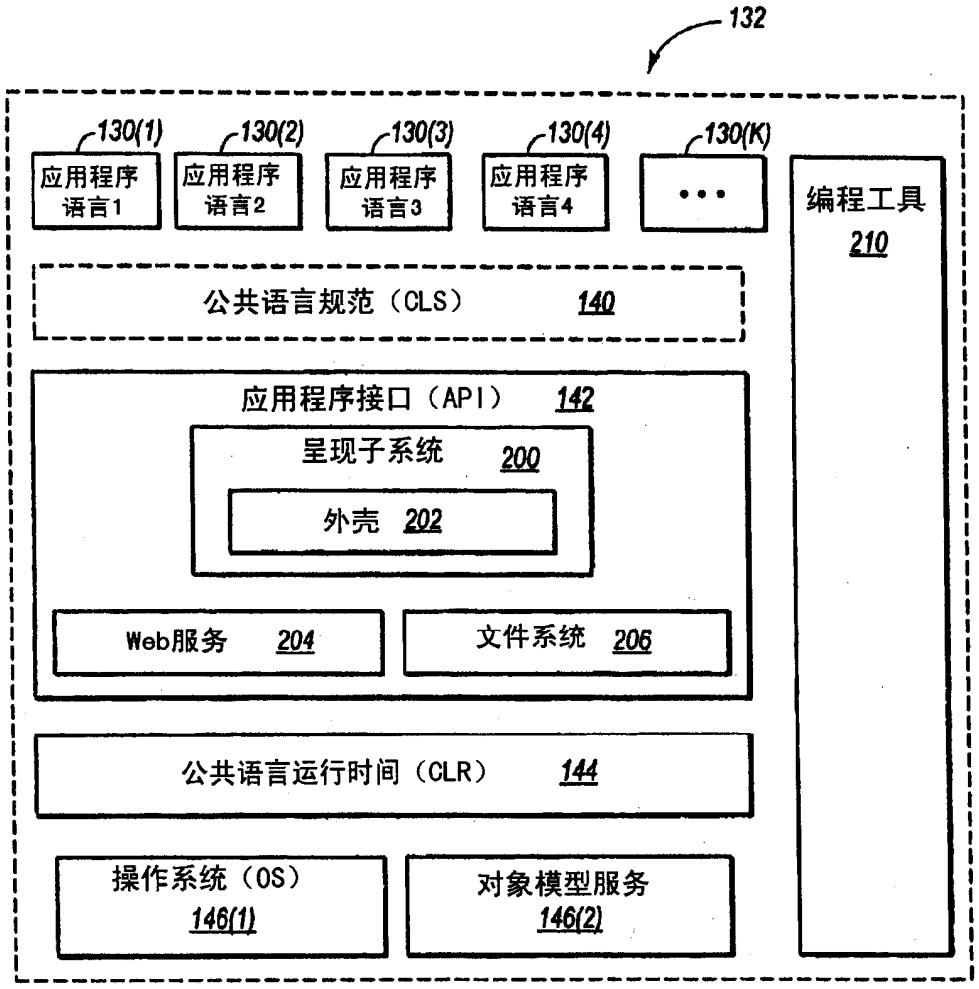

132
130(1)
应用程序语言1
130(2)
应用程序语言2
130(3)
应用程序语言3
130(4)
应用程序语言4
130(K)
...
编程工具
210
公共语言规范（CLS） 140
应用程序接口（API） 142
呈现子系统 200
外壳 202
Web服务 204
文件系统 206
公共语言运行时间（CLR） 144
操作系统（OS）
146(1)
对象模型服务
146(2)

图 2

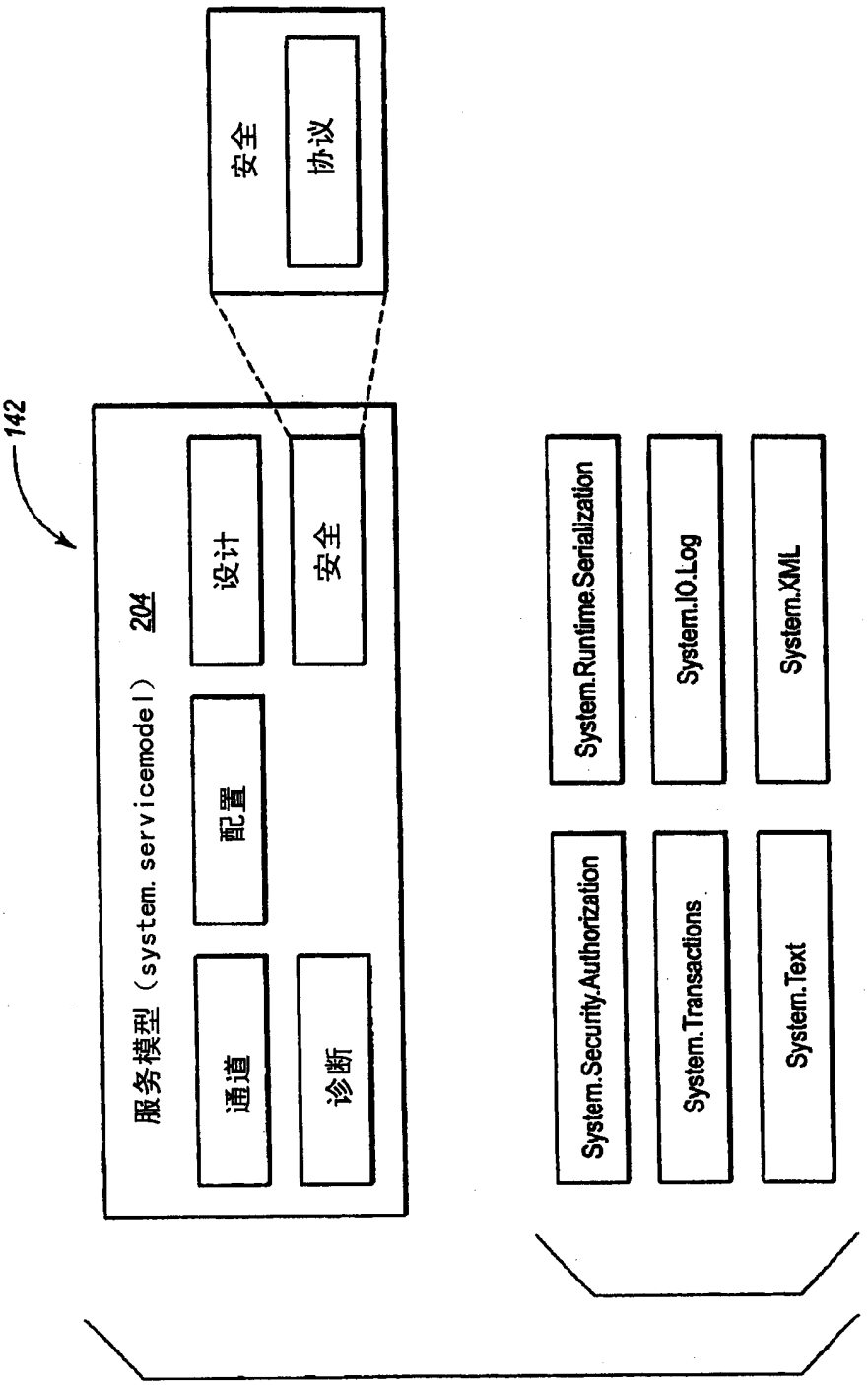

142
服务模型（system.servicemodel） 204
通道
诊断
配置
设计
安全
安全
协议
System.Security.Authorization
System.Transactions
System.Text
System.Runtime.Serialization
System.IO.Log
System.XML

图 3

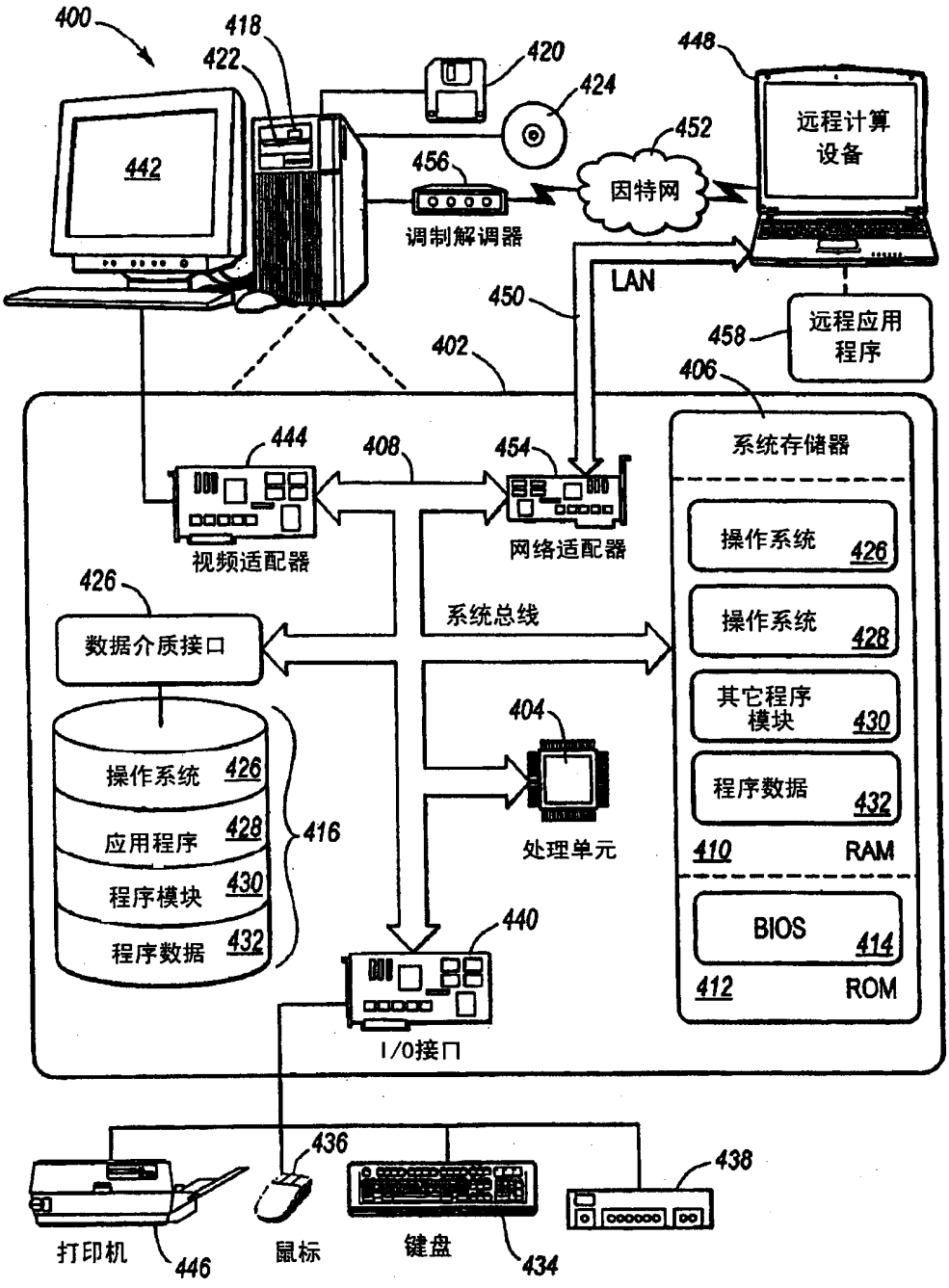

400
418
422
420
424
448
远程计算设备
442
452
因特网
456
调制解调器
LAN
450
远程应用程序
458
402
406
系统存储器
444
408
454
视频适配器
网络适配器
操作系统 426
操作系统 428
其它程序模块 430
程序数据 432
426
数据介质接口
系统总线
404
操作系统 426
应用程序 428
程序模块 430
程序数据 432
416
处理单元
410 RAM
BIOS 414
412 ROM
440
I/O接口
436
438
打印机
446
鼠标
键盘
434

图 4

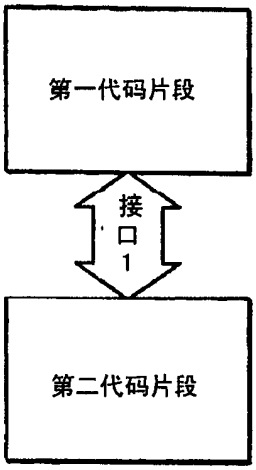

第一代码片段
接口1
第二代码片段

图 5

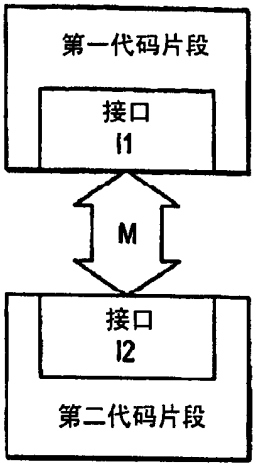

第一代码片段
接口I1
M
接口I2
第二代码片段

图 6

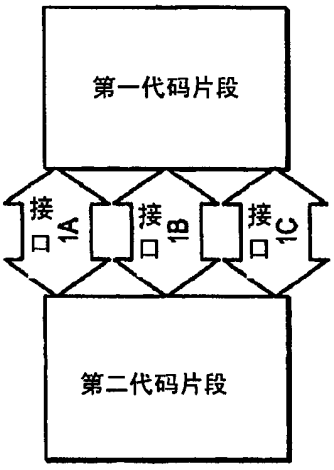

第一代码片段
接口1A
接口1B
接口1C
第二代码片段

图 7

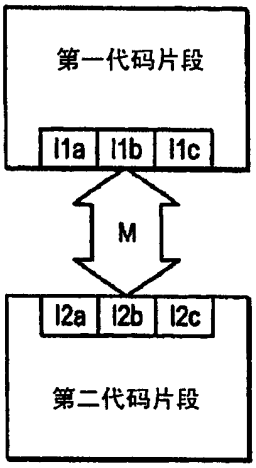

第一代码片段
I1a
I1b
I1c
M
I2a
I2b
I2c
第二代码片段

图 8

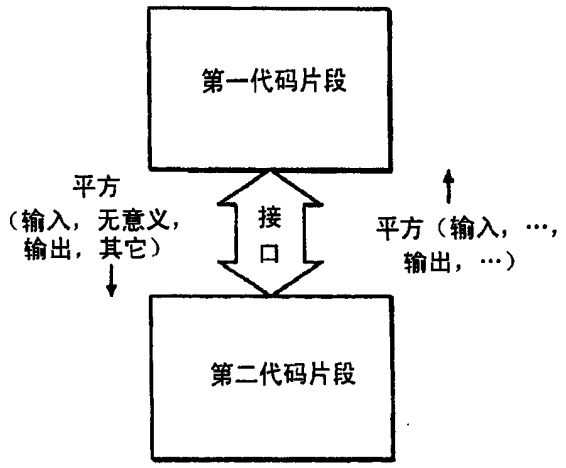


图 9

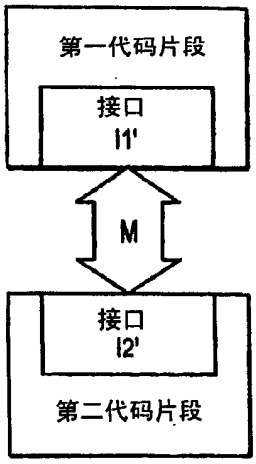


图 10

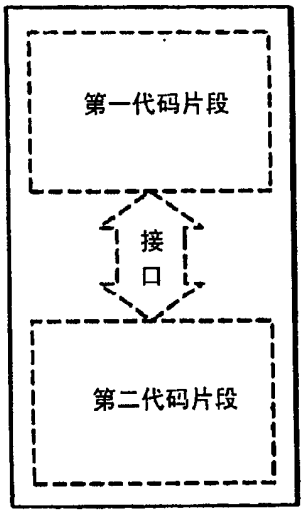


图 11

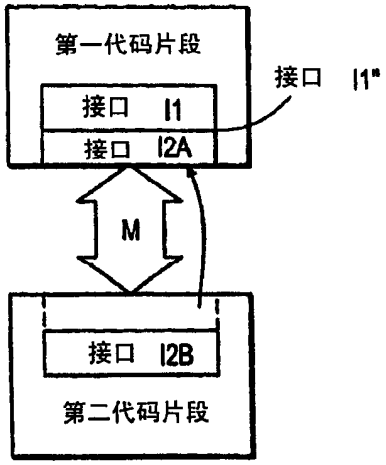


图 12

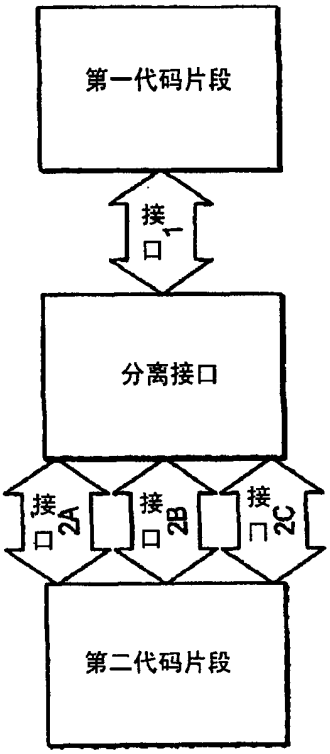

第一代码片段
接口1
分离接口
接口2A
接口2B
接口2C
第二代码片段

图 13

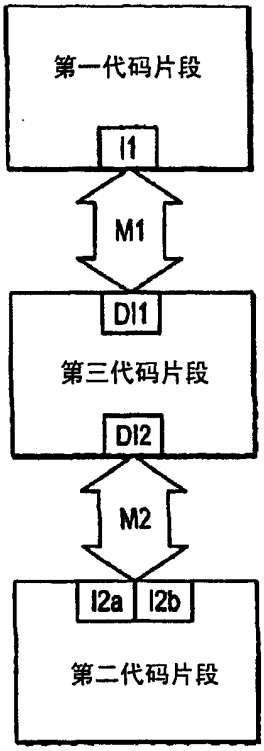

第一代码片段
I1
M1
DI1
第三代码片段
DI2
M2
I2a
I2b
第二代码片段

图 14

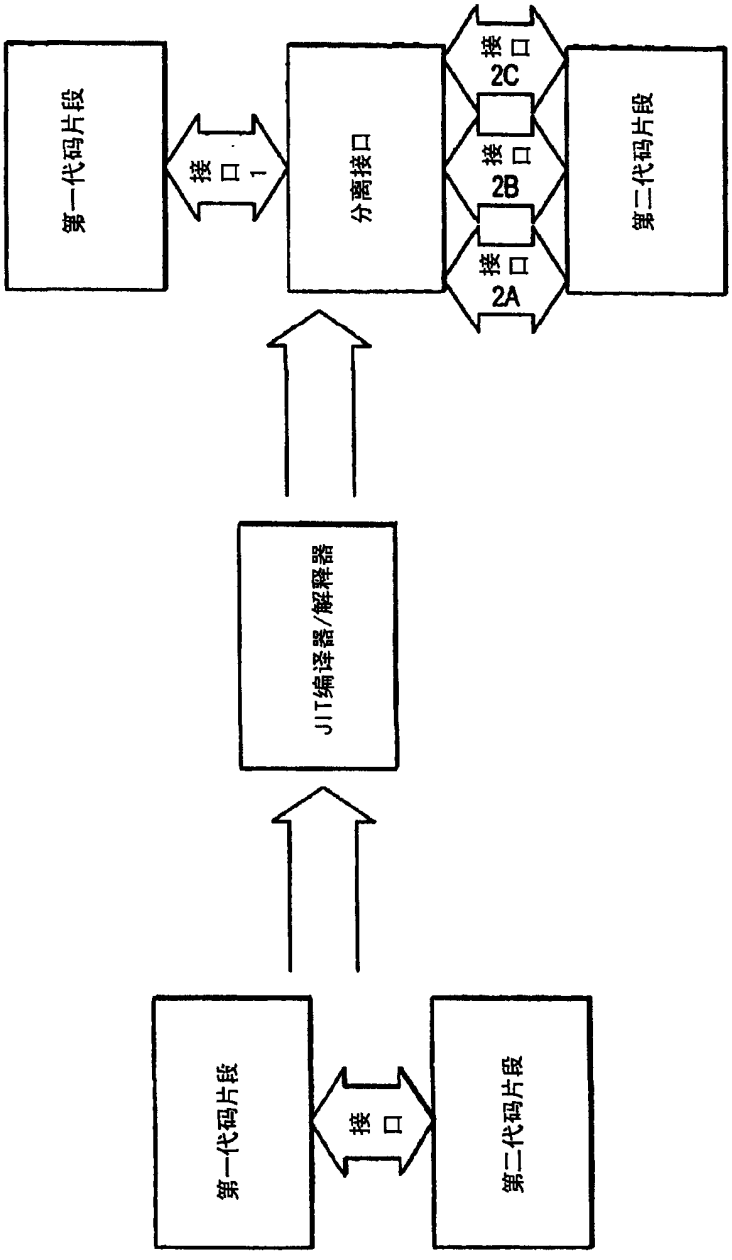

第一代码片段
接口1
分离接口
接口2C
接口2B
接口2A
第二代码片段
JIT编译器/解释器
第一代码片段
接口
第二代码片段

图 15

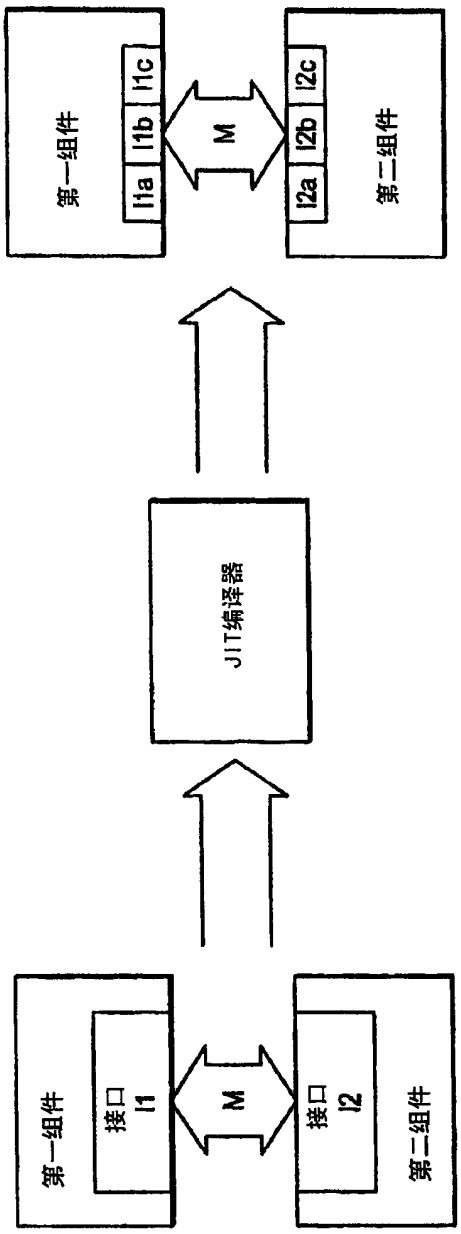

第一组件
接口
I1
M
接口
I2
第二组件
JIT编译器
第一组件
I1a
I1b
I1c
M
I2a
I2b
I2c
第二组件

图 16