



## [12] 发明专利说明书

专利号 ZL 200410078504.1

[45] 授权公告日 2009 年 8 月 19 日

[11] 授权公告号 CN 100530083C

[22] 申请日 2004.9.13

[21] 申请号 200410078504.1

[30] 优先权

[32] 2003.10.24 [33] US [31] 10/694,080

[73] 专利权人 微软公司

地址 美国华盛顿州

[72] 发明人 J·L·伯格丹 R·A·雷耶

[56] 参考文献

US2001/0045961A1 2001.11.29

US6353451B1 2002.3.5

US2002/0054046A1 2002.5.9

US2003/0093419A1 2003.5.15

审查员 杨 薇

[74] 专利代理机构 上海专利商标事务所有限公司

代理人 陈 烽

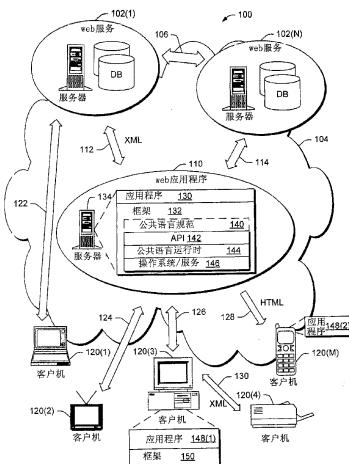
权利要求书 6 页 说明书 121 页 附图 9 页

[54] 发明名称

用于创建应用程序的系统及方法

[57] 摘要

一种编程接口提供生成应用软件、文档、媒体表示和其他内容的函数。这些函数允许开发者获取来自操作系统、对象模型服务或其他系统或服务的服务。



1. 一种用于创建应用程序的方法，包括：

使用第一组服务以用于产生将被包含在一个正被编程的应用程序中的图形组件；

使用第二组服务以用于在正被编程的应用程序中绑定类属性到数据源；及  
使用第三组服务以用于在正被编程的应用程序中格式化内容。

2. 如权利要求 1 所述的方法，其特征在于，所述第一组服务包括确定图  
形组件外观的服务。

3. 如权利要求 1 所述的方法，其特征在于，所述第一组服务包括确定图  
形组件行为的服务。

4. 如权利要求 1 所述的方法，其特征在于，所述第一组服务包括确定图  
形组件安排的服务。

5. 如权利要求 1 所述的方法，其特征在于，所述第一组服务包括多个嵌  
套的定义图形组件的原类型控件。

6. 如权利要求 1 所述的方法，其特征在于，所述图形组件由矢量图形定  
义。

7. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于用动画  
呈现至少一个图形组件的第四组服务。

8. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于创建具  
有浏览能力的应用程序的第四组服务。

9. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于支持电  
子墨水处理系统的第四组服务。

10. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于组合多种不同媒体类型的第四组服务。

11. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于使用浏览器类型的接口在客户机上执行应用程序的第四组服务。

12. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于自动地安装和执行应用程序的第四组服务。

13. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于序列化内容的第四组服务。

14. 如权利要求 1 所述的方法，其特征在于，进一步包括使用用于自动化用户接口的产生的第四组服务。

15. 一种用于创建应用程序的方法，包括：

使用第一组编程服务以在显示内容之前格式化该内容，所述内容将被包含在正被创建的应用程序中；

使用第二组编程服务以在所创建的应用程序中绑定类属性到数据源；及

使用第三组编程服务以产生图像效果，所述图像效果将被包含在所述应用程序中。

16. 如权利要求 15 所述的方法，其特征在于，所述第一组编程服务包括安排多个数据元素。

17. 如权利要求 15 所述的方法，其特征在于，所述第三组编程服务包括用动画呈现至少一个图形项。

18. 如权利要求 15 所述的方法，其特征在于，进一步包括使用要被包含在所述应用程序中的第四组编程服务，其允许应用程序的用户在多个图形之间

---

浏览。

19. 如权利要求 15 所述的方法，其特征在于，进一步包括使用用于编辑被包含在所述被创建的应用程序中的原先创建的内容的第四组编程服务。

20. 如权利要求 15 所述的方法，其特征在于，进一步包括使用用于管理从输入设备接收的输入的第四组编程服务。

21. 如权利要求 15 所述的方法，其特征在于，进一步包括使用用于支持与其他计算系统之间的互操作性的第四组编程服务。

22. 一种用于创建应用程序的方法，包括：

使用第一组服务以产生图形对象以用于正被创建的应用程序中；

使用第二组服务以创建图形对象组件以用于被包括在所创建的应用程序中；及

使用第三组服务以修改图形对象外观。

23. 如权利要求 22 所述的方法，其特征在于，所述第一组服务包括用于定义所创建的应用程序中的至少一个图形对象的行为的服务。

24. 如权利要求 22 所述的方法，其特征在于，所述第一组服务包括用于定义图形对象的安排的服务。

25. 如权利要求 22 所述的方法，其特征在于，所述修改图形对象的外观包括用动画呈现图形对象。

26. 如权利要求 22 所述的方法，其特征在于，所述第二组服务包括用于产生几何形状的服务。

27. 如权利要求 22 所述的方法，其特征在于，进一步包括使用用于格式化文本的第四组服务。

28. 一种用于创建应用程序的方法，其特征在于，所述方法包括：

调用一个或多个第一函数以用于格式化数据，所述数据将被包含在应用程序中；

调用一个或多个第二函数以用于在所述应用程序中创建图形对象；及

调用一个或多个第三函数以用于在所述应用程序中改变图形对象的外观。

29. 如权利要求 28 所述的方法，其特征在于，进一步包括调用一个或多个第四函数以用于使用多个图形对象产生用户接口。

30. 如权利要求 28 所述的方法，其特征在于，进一步包括调用一个或多个第四函数以用于用户接口的运行时创建。

31. 如权利要求 28 所述的方法，其特征在于，进一步包括：

调用一个或多个第四函数以使用多个要被包含在应用程序中的图形对象产生用户接口；及

调用一个或多个第五函数以用于用户接口的运行时创建。

32. 如权利要求 28 所述的方法，其特征在于，所述第一函数帮助：

接收用户输入；及

在显示屏上安排数据元素。

33. 如权利要求 28 所述的方法，其特征在于，所述第二函数被配置成用于产生几何形状。

34. 如权利要求 28 所述的方法，其特征在于，所述第二函数被配置成用于产生至少一个几何形状，且所述第三函数被配置成用于修改所述几何形状的外观。

35. 一种用于创建应用程序的系统，其特征在于，所述系统包括：

用于展现支持创建多个几何形状的第一组函数的装置，从而使得多个几何

---

形状被配置以用于被包含在正被创建的应用程序中；

用于展现支持改变安排几何形状的方式的第二组函数的装置；及

用于展现支持修改几何形状外观的第三组函数的装置。

36. 如权利要求 35 所述的系统，其特征在于，所述第二组函数进一步支持在将呈现的页面上安排几何形状。

37. 如权利要求 35 所述的系统，其特征在于，所述多个几何形状包括直线。

38. 如权利要求 35 所述的系统，其特征在于，所述第三组函数进一步支持关联图像效果和至少一个几何形状。

39. 如权利要求 35 所述的系统，其特征在于，所述第三组函数进一步支持在一段时间上改变特殊几何形状的外观。

40. 如权利要求 35 所述的系统，其特征在于，进一步包括用于展现支持使用多个几何形状产生用户接口的第四组函数的装置。

41. 如权利要求 35 所述的系统，其特征在于，进一步包括用于展现支持关联图形对象和一个或多个数据源的第四组函数的装置。

42. 如权利要求 35 所述的系统，其特征在于，进一步包括用于展现支持显示图形对象特定于数据的版本的第四组函数的装置。

43. 一种用于创建应用程序的方法，其特征在于，所述方法包括：  
调用一个或多个第一函数以用于创建图形对象的组件；  
调用一个或多个第二函数以用于产生图形对象，所述图形对象用于被包含在正被编程的应用程序中；

调用一个或多个第三函数以用于修改图形对象的外观；

调用一个或多个第四函数以用于安排图形对象；及

---

调用一个或多个第五函数以用于关联图形对象和数据源。

44. 如权利要求 43 所述的方法，其特征在于，进一步包括调用一个或多个第六函数以用于在内容的多个显示之间进行浏览。

45. 如权利要求 43 所述的方法，其特征在于，所述图形对象的组件包括多种形状。

46. 如权利要求 43 所述的方法，其特征在于，所述第三函数包括修改特殊图形对象外观的函数。

47. 如权利要求 43 所述的方法，其特征在于，所述第三函数包括修改一个或多个图形对象组件的外观的函数。

48. 如权利要求 43 所述的方法，其特征在于，所述第三函数包括将图形对象移动到显示屏上的不同位置的函数。

49. 如权利要求 43 所述的方法，其特征在于，所述第三函数修改图形对象的外观以响应用户输入。

50. 如权利要求 43 所述的方法，其特征在于，所述第四函数修改图形对象的安排以响应用户输入。

## 用于创建应用程序的系统及方法

### 技术领域

本发明涉及软件和软件开发。更特别地，本发明涉及有助于应用程序和计算机硬件使用软件平台的编程接口。

### 所附光盘简要说明

此说明书所附的三张光盘存储代码名称为“Longhorn”的Microsoft® Windows®操作系统的软件开发工具(SDK)。SDK 包含代码名称为“Longhorn”的 Microsoft® Windows®操作系统的文档。三张光盘的重复拷贝也附在本说明书之后。

三张光盘中的第一张光盘(3 张光盘中的 CD 1)包含 2003 年 10 月 22 日创建的名为“lhsdk”的文件夹。此文件夹大小为 586MB，包含 9,692 个子文件夹，且包含 44,292 个子文件。三张光盘中的第二张光盘(3 张光盘中的 CD 2)包含 2003 年 10 月 22 日创建的名为“ns”的文件夹。此文件夹大小为 605MB，包含 12,628 个子文件夹，且包含 44,934 个子文件。三张光盘中的第三张光盘(3 张光盘中的 CD 3)包含 2003 年 10 月 22 日创建的名为“ns”的文件夹。此文件夹大小为 575MB，包含 9,881 个子文件夹，且包含 43,630 个子文件。三张光盘的每一张上的文件均可以在基于 Windows® 的计算设备(如，IBM-PC 或其等价)上执行，这些计算设备执行 Windows® 品牌的操作系统(如 Windows® NT、Windows® 98、Windows® 2000、Windows® XP 等等)。包含三张光盘的每一张上的文件加入这里作为参考。

三张光盘的每一张都是 CD-R，且符合 ISO 9660 标准。三张光盘的每一张的内容均符合美国信息交换标准码(ASCII)。

### 发明背景

很早以前起，计算机软件就被归类为“操作系统”软件或“应用”软件。广义上来说，应用软件是用于为计算机用户执行特定任务(如解数学方程或支持字处理)的软件。操作系统 是管理和控制计算机硬件的软件。操作系统的

的目标是使得计算机资源对应用程序员可用而同时隐去了实际控制硬件所需的复杂性。

操作系统通过总称为应用编程接口或 API 的函数使得资源可用。术语 API 也被用于指这些的函数中的一个。这些函数通常根据它们对应用程序员提供何种资源或服务来分组。应用软件通过调用单独的 API 函数来请求资源。API 函数也用作将操作系统提供的消息和信息转发回应用软件的方法。

除了硬件的改变，驱动操作系统软件进化的另一个因素是简化和加速应用软件开发的需要。应用软件开发是令人畏惧的任务，有时需要几年的开发者时间来创建有上百万行代码的复杂程序。对流行的操作系统，如各种版本的 Microsoft Windows® 操作系统，应用软件开发者每年编写成千上万个使用此操作系统的不同应用软件。需要严谨的可用操作系统基础来支持如此多的各种应用开发者。

通常，通过使操作系统更加复杂可以使应用软件的开发变得更简单。即，如果一个函数对许多不同的应用程序有用，则最好是写一次并将其包括在操作系统中，而不是要求很多的软件开发者多次编写该函数以包括在很多不同的应用中。以此方式，如果操作系统广泛支持很多应用软件所需的共同功能，则可以在应用软件开发成本和时间上实现显著的节省。

无论操作系统和应用软件之间的界线画在何处，很清楚的是对有用的操作系统来说，在操作系统和计算机硬件及应用软件之间的 API 和操作系统自身有效的内部操作同等重要。

在过去几年中，因特网（总的来说，网络技术）的广泛采用已经改变了计算机软件开发者的眼界。过去，软件开发者着眼于单站点软件应用，以用于独立的桌面计算机，或通过局域网（LAN）连接到有限数量的其他计算机的基于 LAN 的计算机。这样的软件应用通常被称为“紧缩包（shrink wrapped）”的产品，因为这样的软件是在紧缩包的包装中营销和销售的。应用软件使用定义良好的 API 来访问计算机的底层操作系统。

随着因特网的发展并得到了广泛的接受，业界开始认识到在万维网（或简单地“Web”）的各个网站上驻留应用软件的能力。在联网的世界中，来自任何地方的客户机可以向在各个地点驻留的基于服务器的应用提交请求，并在几分之一秒的时间内接收返回的响应。然而，这些 Web 应用软件通常是使用起初为独立的计算设备或本地联网的计算机开发的同一操作系统平台开发

的。不幸的是，在某些情况下，这些应用不能充分迁移到分布式计算领域。底层平台就不是用支持无限数量互连计算机的思想来构建的。

为了适应由因特网带来的到分布式计算环境转变，微软公司开发了称为“.NET”框架(读做“点 Net”)的网络软件平台。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在因特网上执行的 Web 服务。用于微软.NET™ 框架的一组 API 函数伴随着这样的动态转变。

随着.NET™ 框架的使用变得越来越常见，已建立了提高此平台效率和/或性能的多种方法。发明人已开发出一组独特的 API 函数来应对提高的效率和/或性能。

### 发明概要

一种编程接口提供生成应用软件、文档、媒体表示和其他内容的函数。这些函数允许开发者获取来自操作系统、对象模型服务或其他系统或服务的服务。在一个实施例中，这些函数允许开发者生成图形用户接口。

### 附图说明

相同的编号在整个附图中被用来引用类似的部件。

图 1 展示网络架构，在其中客户机使用常规协议通过因特网访问 Web 服务；

图 2 是用于网络平台的软件架构的方框图，它包括应用编程接口 (API)；

图 3 是由 API 及各种 API 函数的功能分类支持的表示子系统的方框图；

图 4 是可以执行所述软件架构的全部或部分的示范计算机的方框图；及

图 5、6、7、8、9、10、11、12、13、14、15 和 16 展示编程接口的各种实施例。

### 详细说明

本发明涉及用于开发者可以在其上构建 Web 应用和服务的网络平台的应用编程接口 (API)。更特别地，所描述的示范 API 是对使用网络平台的操作系统，如微软公司创建的.NET™ 框架的操作系统。.NET™ 框架是在分布式计算环境中实现的 Web 服务和 Web 应用软件的软件平台。它使用开放的通讯标准，在合作执行特殊任务的松散耦合的 Web 服务之间通讯，代表下一代的因特网

计算。

在上述实施例中，网络平台使用 XML(可扩展标记语言)，一种用于描述数据的开放标准。XML 由万维网联盟(W3C)管理。XML 被用于定义网页和企业对企业文档上的数据元素。XML 使用类似于 HTML 的标签结构，然而，HTML 定义如何显示元素，而 XML 定义那些元素包含什么。HTML 使用预定义的标签，而 XML 允许标签由页面的开发者来定义。这样，实际上可以识别任何数据项，允许 Web 页面起着类似数据库记录的作用。通过使用 XML 和其他开放协议，如简单对象访问协议(SOAP)，网络平台允许集成可以根据用户需要定制的各种服务。虽然，结合 XML 和其他开放标准来说明在此所述的实施例，这对本发明的操作不是必须的。

如在此所用，短语应用编程接口或 API 包括使用方法或函数调用及远程调用(如，代理、占位程序(stub)关系)和 SOAP/XML 调用的传统接口。

应理解，在下面的一些名字空间说明中，特定类、接口、枚举和代理的说明是空的。这些类、接口、枚举和代理更完全的说明可以在上面引用的存储 SDK 的光盘的主题中找到。

### 示范网络环境

图 1 展示网络环境 100，在其中可以实现网络平台，如.NET<sup>TM</sup>框架。网络环境 100 包括代表性的 Web 服务 102(1), ..., 102(N)，它们提供可以通过网络 104(如，因特网)访问的服务。Web 服务(总地使用编号 102 来引用)是可编程的应用组件，它们可以重用并通过网络 104 以程序方式交互，这种交互通常通过工业标准的 Web 协议进行，如 XML、SOAP、WAP(无线应用协议)、HTTP(超文本传输协议)和 SMTP(简单邮件传输协议)，虽然也可以使用通过网络和 Web 服务交互的其他方法，如远程过程调用(RPC)或对象代理类型技术。Web 服务可以是自我描述的并通常根据消息的格式和顺序来定义。

Web 服务 102 可以由其他服务(如通讯链接 106 所示)或软件应用，如 Web 应用 110(如通讯链接 112 和 114 所示)直接访问。所示的每个 Web 服务 102 包括一个或多个服务器，这些服务器执行处理对特殊服务的请求的软件。这样的服务通常维护存储返回给请求者的信息的数据库。Web 服务 可以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校验、通知、数据库存储、股票报价、位置目录、映射、音乐、电子钱包、日历/时间表、列表、

---

电话收听、新闻和信息、游戏、票务等等。Web 服务可以互相和与其他应用合，并来构建智能的交互操作。

网络环境 100 也包括使用 Web 服务 102(如通讯链接 122 所示)和/或 Web 应用 110(如通讯链接 124、126 和 128 所示)的代表性的客户机设备 120(1)、120(2)、120(3)、120(4)、...、120(M)。客户机也可以使用标准协议互相通讯，如客户机 120(3)和 120(4)之间的示范 XML 链接 130 所示。

客户机设备(总地使用编号 120 来引用)可以用很多不同的方法来实现。可能的客户机实现的例子包括，而不限制，便携式计算机、固定计算机、输入板 PC、电视机/机顶盒、无线通讯设备、个人数字助理、游戏终端、打印机、复印机和其他智能设备。

Web 应用 110 是一应用程序它设计成在网络平台上运行，并可以在对来自客户机 120 的请求进行处理和服务时使用 Web 服务 102。Web 应用 110 由在编程框架 132 之上运行的一个或多个软件应用 130 组成，编程框架 132 在一个或多个服务器或其他计算机系统上执行。注意，Web 应用 10 的一部分可以实际上驻留一个或多个客户机 120 上。另外，Web 应用 110 可以和客户机 120 上的其他软件协调来实际完成它的任务。

编程框架 132 是支持由应用开发者开发的应用软件和服务的结构。它通过支持多种语言，允许多种语言的开发和无缝集成。它支持开放的标准，如 SOAP，并封装底层的操作系统和对象模型服务。框架对多种编程语言提供健壮和安全的执行环境并提供安全、集成的类库。

框架 132 是包含应用编程接口 (API) 层 142、通用语言运行库 (CLR) 层 144 和操作系统/服务层 146 的多层架构。这种分层的架构允许对各层进行更新和修改而不影响框架的其他部分。通用语言规范 (CLS) 140 允许各种语言的设计者编写可以访问底层库功能的代码。规范 140 起语言设计者和库设计者之间的合约的作用，可用于提升语言互操作性。通过遵从 CLS，以一种语言编写的库可以由以其他语言编写的代码模块直接访问，从而获得以一种语言编写的代码模块和以另一种语言编写的代码模块之间的无缝集成。CLS 的一个详细实施例在由参与者在 ECMA TC39/TG3 中创建的 ECMA 标准中说明。读者请参考 ECMA 在 [www.ecma.ch](http://www.ecma.ch) 的网站。

API 层 142 表示应用 130 可以调用它们以访问层 146 提供的资源和服务的函数组。通过对网络平台展现 API 函数，应用软件开发者可以创建用于分布

式计算系统的 Web 应用软件以充分利用网络资源和其他 Web 服务，而无需理解实际上如何操作或提供那些网络资源的复杂内部机制。再者，Web 应用软件可以用任何数量的编程语言来编写，并转换为由通用语言运行库 144 支持并作为通用语言规范 140 的部分包括的中间语言。以此方式，API 层 142 可以为广泛的不同应用提供各种方法。

另外，框架 132 可以配置为支持由通过容纳框架的服务器 134 远程执行的远程应用软件进行的 API 调用。代表性应用软件 148(1)和 148(2)各自驻留在客户机 120(3)和 120(M)上，它们可以通过网络 104 直接或间接调用 API 层来使用 API 函数。

框架也可以在客户机上实现。客户机 120(3)表示其中框架 150 在客户机上实现的情况。此框架可以等同于基于服务器的框架 132 或根据客户机的目的进行修改。另外，基于客户机的框架可以在客户机为有限功能或专用功能的设备，如手机、个人数字助理、手持计算机或其他通讯/计算设备的情况下进行精简。

### 开发者的编程框架

图 2 更详细地展示编程框架 132。通用语言规范 (CLS) 层 140 支持以各种语言 130(1)、130(2)、130(3)、130(4)、...、130(K) 编写的应用软件。这样的应用语言包括 Visual Basic、C++、C#、COBOL、Jscript、Perl、Eiffel、Python 等等。通用语言规范 140 规定特性的子集或有关特性的规则，若遵从这些规则，则允许各种语言进行通讯。例如，某些语言不支持可能由通用语言运行库 144 支持的特定类型（如，“int\*” 类型）。在此情况，通用语言规范 140 不包括该类型。另一方面，由所有或大多数语言支持的类型（如“int[]” 类型）被包括在通用语言规范 140 中，从而库开发者可以不必使用它而确保那些语言可以处理它。此通讯能力带来以一种语言编写的代码模块和以另一种语言编写的代码模块之间的无缝集成。由于不同的语言特别适合于各种特殊的任务，语言之间的无缝集成允许开发者对特殊的代码模块选择特殊的语言，且以不同语言编写的模块能够使用该代码模块。通用语言运行库 144 通过跨越语言继承支持无缝的多语言开发，并对多种编程语言提供健壮和安全的执行环境。关于通用语言规范 140 和通用语言运行库 144 的更多信息，读者请参考一起待批准的 2000 年 6 月 21 日提交、标题为“编译多种语言的方

法和系统”的专利申请(序列号 09/598, 105)和 2000 年 7 月 10 日提交、标题为“统一数据类型系统和方法”的申请(序列号 09/613, 289)，将它们包含在此作为参考。

框架 132 封装操作系统 146(1)(如 Windows® 品牌的操作系统)和对象模型服务 146(2)(如组件对象模型(COM)或分布式 COM)。操作系统 146(1) 提供常规函数，如文件管理、通知、事件处理、用户接口(如，窗口、菜单、对话框等等)、安全、身份验证、校验、进程和线程、内存管理等等。对象模型服务 146(2) 提供连接其他对象的接口来执行各种任务。对 API 层 142 做出的调用被传递给通用语言运行库层 144，以由操作系统 146(1) 和/或对象模型服务 146(2) 在本地执行。

API 142 将 API 函数分组到多个名字空间中。名字空间本质上定义类、接口、代理、枚举和结构的集合，它们统称为“类型”并提供相关功能的特定集合。类表示管理下的堆分配的数据，它具有引用赋值语义。代理是面向对象的函数指针。枚举是特殊类型的表示命名常量的值类型。结构表示静态分配的数据，它具有值赋值语义。接口定义其他类型可以实现的合约。

通过使用名字空间，设计者可以将一组类型组织成分层的名字空间。设计者能够从该组类型创建多个组，每组至少包括一个逻辑上显现出相关的功能的类型。在示范实施例中，API 142 组织为包括三个根名字空间。应注意，虽然在图 2 中只展示了三个根名字空间，也可以在 API 142 中包括另外的根名字空间。API 142 中展示的三个根名字空间是：用于表示子系统的第一个名字空间 200(包括用于用户接口外壳的名字空间 202)，用于 Web 服务的第二个名字空间 204，及用于文件系统的第三个名字空间 206。每个组可以分配一个名字。例如，表示子系统名字空间 200 中的类型可以分配名字“Windows”，而文件系统名字空间 206 中的类型可以分配名字“Storage”。命名的组可以在用于系统层 API 的单个“全局根”名字空间，如总系统名字空间下面进行组织。通过选择并加上顶层标识符前缀，每个组中的类型可以容易地通过层次名字来引用，层次名字包含所选择的加在包含该类型的组名字前面的顶层标识符前缀。例如，文件系统名字空间 206 内的类型可以使用层次名字“System.Storage”来引用。以此方式，各个名字空间 200、204 和 206 成为系统名字空间的主要分支，并可以带有这样的标志，其中各个名字空间用命名符加上前缀，如前缀“System.”。

表示子系统名字空间 200 涉及编程和内容开发。它提供用于处理应用软件、文档、媒体表示和其他内容的产生的类型。例如，表示子系统名字空间 200 提供允许开发者获取来自操作系统 146(1) 和/或对象模型服务 146(2) 的服务的编程模型。

外壳名字空间 202 涉及用户接口功能。它提供允许开发者在他们的应用中嵌入用户接口功能的类型，并进一步允许开发者扩展用户接口功能。

Web 服务名字空间 204 涉及支持创建各种应用的架构，这些应用包括如象在内联网上的两个伙伴之间操作的聊天应用那样简单的应用，和/或象用于成百万用户的可伸缩 Web 服务那样复杂的应用。由于只需要使用适合于特殊解决方案复杂度的那些部分，上述架构最好是高度可变。该架构提供构建各种规模和复杂度的基于消息的应用的基础。该架构或框架提供用于基本消息、安全消息、可靠消息和事务消息的 API。在下述实施例中，将相关 API 以精心设计来平衡实用性、可用性、可扩展性和版本可管理性的方式分解为名字空间的层次。

文件系统名字空间 206 涉及存储。它提供用于信息存储和检索的类型。

除了框架 132 之外，提供编程工具 210 来辅助开发者构建 Web 服务和/或应用。编程工具 210 的一个例子是 Visual Studio™，它是由微软公司提供一种多语言编程工具套件。

### 根 API 名字空间

图 3 更详细地展示表示子系统 200 的一部分。在一个实施例中，根据在其中周期地连接名字字符串的层次命名协定来识别名字空间。例如，通过根名字 “System.Windows” 来识别表示子系统名字空间。在 “System.Windows” 名字空间内有另一个用于各种控件的名字空间，称为 “System.Windows.Controls”，它进一步标识称为 “System.Windows.Controls.Primitives”的另一个用于原语的名字空间(未标出)。记住了这样的命名协定，下面提供 API 142 所选择的名字空间的概要，虽然使用其他命名协定能产生同等效果。

如图 3 所示，表示子系统 200 包括多个名字空间。图 3 所示的名字空间所表示的是表示子系统 200 的特殊实施例。表示子系统 200 的其他实施例可以包括一个或多个另外的名字空间或可以忽略图 3 所示名字空间中的一个或

多个。

表示子系统 200 是用于 API 142 的很多表示功能的根名字空间。控件名字空间 310 包括用来构建信息显示的控件，如用户界面，及允许用户与应用交互的类。例子控件包括在显示屏上创建按钮的“Button”、在显示屏上生成单选按钮的“RadioButton”，在显示屏上创建菜单的“Menu”、在显示屏上创建工具栏的“ToolBar”、在显示屏上生成图形的“Image” 及创建信息的层次视图的“TreeView”。

通过嵌套和摆放多个元素可以创建某些的控件。控件具有隐藏用于创建控件的元素的逻辑模型，从而简化编程模型。可以由开发者或用户(如，通过客户在用户界面按钮的外观和行为)设置控件的样式和主题。某些控件具有允许个人调整单个控件的样式的可定位组件。另外，由应用开发者和组件开发者可以生成控件的子类并扩展控制。使用矢量图来呈现控件使得可以重新设置它们的大小来适应特殊接口或其他显示的要求。控件能够使用动画来增强例如用户接口的交互感觉并显示动作和反应。

### 控件名字空间

控件名字空间 310 包括一个或多个面板，面板是测量和安排其子控件(如，嵌套的元素)的控件。例如，“DockPanel”面板通过将每个子控件泊放到显示的上边、左边、下边或右边，并用其他数据填充余下的空间来安排子控件。例如，特殊的面板可以将菜单和工具栏泊放到显示的上边、状态栏泊放到显示的下边、文件夹列表泊放到显示的左边，并用消息列表填充余下的空间。

上面提到，`System.Windows.Controls.Primitives` 是包括多个控件的名字空间，这些控件是通常由在 `System.Windows.Controls` 名字空间中的控件的开发者以及由创建他们自己的控件的开发者使用的组件。这些组件的例子包括“Thumb 和 RepeatButton”。另一个组件“ScrollBar”是使用四个重复按钮(一个用于“向上一行”、一个用于“向下一行”、一个用于“向上翻页”，一个用于“向下翻页”)和“Thumb”来创建的，用于拖动当前视图到文档中另一位置的。在另一个例子中，“ScrollViewer”是使用两个“ScrollBars”和一个“ScrollArea”创建以提供可滚动区域的控件。

下面的列表包括由 `System.Windows.Controls` 名字空间显现的例子类。

例如，这些类允许用户和应用通过各种输入和输出能力以及附加的显示能力进行交互。

- AccessKey — AccessKey 是包装一个字符的 FrameworkElement 元素，表示将把表示字符的键盘记号装饰作为键盘助记符来接收。缺省的键盘记号装饰为下划线。
- Audio — 音频元素。
- Border — 在另一个元素周围画出边界、背景或两者。
- Button — 表示内在地响应点击(Click)事件的标准按钮组件。
- Canvas — 定义在其中用户可以通过相对于 Canvas 区域的坐标明确地定位子元素的区域。
- CheckBox — 使用 CheckBox 向用户给出如真/假这样的选项。CheckBox 允许用户从选项列表中进行选择。CheckBox 控件使用户选择选项的组合。
- CheckedChangedEventArgs — 这个类包含有关 CheckedChangeEvent 事件的附加信息。
- CheckStateChangedEventArgs — 这个类包含有关 CheckStateChangeEvent 事件的附加信息。
- ClickEventArgs — 包含有关 Click 事件的信息。
- ColumnStyle — 表示可改变的 ColumnStyle 对象。
- ColumnStyles — 可改变模式的 IList 对象，它为可改变的(Changeable)元素的集合。
- ComboBox — ComboBox 控件。
- ComboBoxItem — 实现可在 ComboBox 内可选的项目的控件。
- ContactPickerDialog — 允许用户选择一个或多个联系。
- ContactPropertyRequest — 允许应用通过 ContactPickerDialog 请求有关联系属性的信息。此类不能被继承。
- ContactPropertyRequestCollection — 表示 ContactPropertyRequest 对象的集合。
- ContactSelection — 有关来自代码名称为“WinFS”的 Microsoft® Windows® 文件系统或 Microsoft Active Directory®(活动目录®)的所选择的联系的信息。

- ContactSelectionCollection — 表示 ContactSelection 对象的集合。
- ContactTextBox — 支持选取联系或联系的属性的编辑控件。
- ContactTextBoxSelectionChangedEventArgs — ContactTextBoxSelectionChanged 事件的参数。
- ContactTextBoxTextChangedEventArgs — ContactTextBoxTextChanged 事件的参数。
- ContactTextBoxTextResolvedEventArgs — TextResotvedToContact 事件的参数。
- ContentChangedEventArgs — ContentChangedEvent 的事件参数。
- ContentControl — 具有单条内容的所有控件的基类。
- ContentPresenter — 当添加内容时，在内容控件的样式之内使用 ContentPresenter 来表示在控件的可视树中的位置(色彩模板)。
- ContextMenu — 定义菜单的控件，菜单中包含用户将引用的选择。
- ContextMenuEventArgs — 在 ContextMenuEvent 事件发生时发送的数据。
- Control — 表示对所有用户交互元素的基类。此类向其子类提供基本的属性集合。
- Decorator — 将效果应用到单个子元素之上或应用到单个子元素周围的元素(如边界 Border)的基类。
- DockPanel — 定义用户在其中可以彼此水平或垂直地排放子元素的区域。
- DragDeltaEventArgs — 此类包含有关 DragDeltaEvent 事件的附加信息。
- FixedPanel — FixedPanel 是在固定格式文档中使用以便包含用于分页的固定的页根元素。FixedPanel 一次显示分页内容的一页，或将其作为可滚动的页堆栈。
- FlowPanel — FlowPanel 用于划分、缩进和对齐超出单行长度的内容。FlowPanel 提供分行和对齐属性，它能在容件的内容流，例如文本，可能超出单行长度时使用。
- Frame — 可以加载另一个标记树内容的区域。

- Generator — Generator 是为 ItemsControls 生成 UI 的对象，它在 GeneratorFactory 的监控下工作。
- GeneratorFactory — GeneratorFactory 负责为 ItemsControl 生成 UI。它维护控件的 ItemsCollection(扁平视图)中的项与对应的 UIElement 之间的关联。该控制的项一容件能向 Factory 请求实际产生 UI 的的生成器(Generator)。
- GridPanel — 定义包含列和行的网格区域。
- HeaderItemsControl — 包含多个项目且具有标题的所有控件的基类。
- HorizontalScrollBar — 水平滚动条(ScrollBar)类。
- HorizontalSlider — 水平滑动器(Slider)类。
- HyperLink — HyperLink 类实现浏览控制。缺省的表示器为 TextPresenter。
- Image — 提供在文档或应用中包含图像的简单方法。
- IncludeContactEventArgs — 传递给 ContactPickerDialog. IncludeContact 事件的处理程序的参数。
- ItemCollection — 维护控件内离散的项的集合。提供支持改变集合内容及获取有关内容的数据的方法和属性。
- ItemsChangedEventArgs — 由 GeneratorFactory 触发 ItemsChanged 事件以通知布局组件，该项集合已被更改。
- ItemsControl — 有多个子控件的所有控件的基类。
- ItemsView — ItemsView 提供 ItemCollection 的平视图。
- KeyboardNavigation — KeyboardNavigation 类提供在所关注的控件之间进行逻辑(Tab 键)和方向性(箭头)浏览的方法。
- ListBox — 实现可选择的项列表的控件。
- ListItem — 实现 ListBox 内可选择的项的控件。
- Menu — 定义供用户调用的选项的菜单的控件。
- MenuItem — Menu 的子项。可以选择 MenuItem 来调用命令。MenuItem 可以为分隔符。MenuItem 可以为子菜单的标题。可以选定或不选定 MenuItem。
- PageViewer — 表示包含分页控件、工具栏和页面栏控件的文档查看复合控件。

- PaginationCompleteEventArgs — PaginationCompleteEvent 的事件参数。
- PaginationProgressEventArgs — PaginationProgressEvent 的事件参数。
- Pane — 提供以标记语言定义窗口属性(如，“XAML”)而不打开新窗口的方法。
- Panel — 对所有的 Panel 元素提供基类。为了实例化 Panel 元素，使用导出的具体类。
- RadioButton — RadioButton 实现有两种状态(真或假)的选项按钮。
- RadioButtonList — 该控件用作组合 RadioButton 的控件并且是处理 RadioButton 互斥性的控件。该 RadioButtonListSelector 继承。该 RadioButtonList 本质上是 SingleSelectionSelector，且 Selection 的概念(来自 Selecton)是断切它不组合的 RodioBntton 的 Check 的属性
- RowStyle — 可改变的模式 Changeable 元素。
- RowStyles — 可改变的模式 IList 对象，它是 Changeable 元素的集合。
- ScrollChangeEventArgs — ScrollChangeEventArgs 描述以滚动状态的变化。
- ScrollViewer —
- SelectedItemsCollection — 对 Selector 中所选择的项的容件。
- SelectionChangedEventArgs — 对选择改变事件处理程序的输入。
- SimpleText — SimpleText 是轻量级的、多行的、单一格式文本元素，用在用户接口(UI)情况中。SimpleText 显现几个和 Text 相同的格式化属性，且通常可以用于以降低通用性为低价来换取性能提高。
- StyleSelector — StyleSelector 允许应用软件编写者提供客户风格的式选择逻辑。例如，当类 Bug 作为内容时，对 Pri1 错误使用特殊样式而对 Pri2 错误使用不同的样式。应用软件编写者可以覆盖导出的选择器类中的 SelectStyle 方法，并将此类的实例赋值到 ContentPresenter 类上的 StyleSelector 属性。
- Text — 表示支持呈现多种格式文本的文本控件。Text 最好在应用 UI

内使用；更高级的文本情况得益于 `TextPanel` 的附加特性集合。在多数需要相对简单的文本支持的情况下，由于它轻量级的本质和特性的范围，`Text` 是首选的元素。

- `TextBox` — 表示提供接受文本输入的可编辑区域的控件。
- `TextChangedEventArgs` — `TextChangedEventArgs` 表示与由 `TextRange.SetText()` 触发的事件相关的 `RoutedEventArgs` 的一类型。
- `TextPanel` — 格式化文本、改变文本大小并画出文本。`TextPanel` 支持多行文本和多行文本格式。
- `ToolTip` — 当用户悬停在控件上方时显示信息的控件。
- `ToolTipEventArgs` — `ToolTipEvent` 发生时发送的数据。
- `TransformDecorator` — `TransformDecorator` 包含子控件并对其应用指定的转换。`TransformDecorator` 实现在其本地(转换前)坐标中测量和安排子控件的逻辑，以使得在转换之后子控件紧密适合装饰器的空间并使用最大的区域。因而该子控制不需要知道已对它应用转换的知识。
- `UIElementCollection` — `UIElementCollection` 是 `UIElement` 的有序集合。
- `ValueChangedEventArgs` — `ValueChangedEventArgs` 类包含有关 `ValueChangeEvent` 事件的附加信息。
- `VerticalScrollBar` — 垂直滚动条类。
- `VerticalSlider` — 垂直滑动器类。
- `Video` — 在当前的用户坐标系统内的指定矩形中播放视频流或音频流文件。
- `VisibleChangedEventArgs` — `VisibleChangedEventArgs` 类包含有关 `VisibleChangeEvent` 事件的附加信息。

`System.Windows.Controls` 名字空间也包含各种枚举。下面的列表包含和 `System.Windows.Controls` 名字空间关联的例子枚举。

- `CharacterCase` — 当键入文本时指定 `TextBox` 控件中字符的大小写。
- `CheckState` — 指定如复选框这样可以选定、不选定或设置为中间状态的控件的状态。

- ClickMode — 指定何时触发 Click 事件。
- ContactControlPropertyPosition — 控制联系的属性的位置和显示。
- ContactPickerDialogLayout — 指定 ContactPickerDialog 如何显示选择的属性。
- ContactPropertyCategory — 在属性有用户可以选择的多个值的情况下，指定将哪个值作为缺省值。例如，当从 ContactPickDialog 请求电话号码，且用户选择工作及家庭电话号码作为联系时，若指定“Work-工作”作为优先类别，工作电话号码作为缺省选择出现。用户随后能使用 UI 另外选择家庭电话号码。
- Contact.PropertyType — 指定 ContactPickerDialog 会向用户请求的联系的属性。
- ContactType — 指定哪个联系的类型在 ContactPickDialog 吕显示。
- Direction — 此枚举由 GeneratorFactory 和 Generator 使用来指定生成器产生 UI 的方向。
- Dock — 指定子元素在 DockPanel 内的泊放位置。
- GeneratorStatus — 此枚举由 GeneratorFactory 使用以表示其状态。
- KeyNavigationMode — TabNavigation 属性的类型指定容器在发生 Tab 导航时如何移动着眼点。
- MenuItemBehavior — 定义 MenuItem 可以具有的不同行为。
- MenuItemType — 定义 MenuItem 的不同放置类型。
- Orientation — 滑动器方向类型。
- PageViewerFit — 选择页如何拟合到 PageViewer 的客户区域中。
- PageViewerMode — 在下拉模式列表中选择当前反映的 PageViewer 模式。
- ScrollerVisibility — ScrollerVisibility 定义滚动条的可视行为。
- SelectionMode — 指定 ListBox 的选择行为。

“Position”是和 System.Windows.Controls 名字空间关联的例子结构。Generator 的用户使用此结构描述位置。例如，为了从项列表的头正向开始生成，指定位置 (-1, 0) 和方向 Forward。为了从列表的末端反向开始生成，指定位置 (-1, 0) 和方向 Backward。为了在有下标 k 的元素之后生成项，指定位

置(k, 0)和方向 Forward。

下面的列表包含和 System.Windows.Controls 名字空间关联的例子代理。

- CheckedChangedEventHandler — 此代理由 CheckedChangedEventArgs 事件的处理程序使用。
- CheckStateChangedEventHandler — 此代理由 CheckStateChangedEvent 事件的处理程序使用。
- ClickEventHandler — 表示处理 Click 事件的方法。
- ContactTextBoxSelectionChangedEventHandler — ContactTextBoxSelectionChanged 事件的代理处理程序。
- ContactTextBoxTextChangedEventHandler — ContactTextBoxTextChanged 事件的代理处理程序。
- ContactTextBoxTextResolvedEventHandler — TextResolvedToContact 事件的代理处理程序。
- ContentChangedDelegate — ContentChangedEventArgs 事件的代理。
- ContextMenuEventHandler — 用于处理 ContextMenuEventArgs 事件的回调用类型。
- DragDeltaEventHandler — 此代理由 DragDeltaEventArgs 事件的处理程序使用。
- IncludeContactEventHandler — ContactPickerDialog. IncludeContact 事件的处理程序。
- ItemsChangedEventHandler — 由接收 ItemsChangedEventArgs 的处理程序使用的代理。
- OpenedEventHandler — ContactPickerDialog. Opened 事件的处理程序。
- PaginationCompleteDelegate — PaginationCompleteEventArgs 事件的代理。
- PaginationProgressDelegate — PaginationProgressEventArgs 事件的代理。
- ScrollChangeEventHandler — 此代理由 ScrollChangeEventArgs 事件的处理程序使用。

- SelectionChangedEventHandler — 用于处理选择改变事件的代理类型。
- TextChangedEventHandler — 由接收 TextChangedEventArgs 的处理程序使用的代理。
- ToolTipEventHandler — 用于处理 ToolTipEvent 事件的回调用类型。
- ValueChangedEventHandler — 此代理由 ValueChangedEvent 事件的处理程序使用。
- VisibleChangedEventHandler — 此代理由 VisibleChangedEvent 事件的处理程序使用。

另一个名字空间， System.Windows.Controls.Atoms ，是 System.Windows.Controls 名字空间的子名字空间。 System.Windows.Controls.Atoms 包括关联的控件、事件参数和事件处理程序。下述列表包含和 System.Windows.Controls.Atoms 名字空间关联的例子类。

- PageBar — 表示可滚动的分页控件。
- PageElement — 呈现分页内容的特定页。被呈现的页由 PageSource 属性指定。
- PageHoveredEventArgs — PageHoveredEventArgs 提供有关鼠标指针悬停在何处的信息。
- PageScrolledEventArgs — PageScrolledEventArgs 包含涉及 PageScrolled 事件的信息。
- PageSelectedEventArgs — 当选择新的行 / 列范围时触发 PageSelectedEvent 事件。
- PageSelector — PageSelector: 允许用户选择将被显示的页的行 / 列范围。
- PageSource — 识别将被分页的内容来源。它也提供用于格式化分页内容的属性和方法。

下述列表包含和 System.Windows.Controls.Atoms 名字空间关联的例子代理。

- PageHoveredEventHandler — 此代理由 PageHoveredEvent 事件的处理器使用。
- PageScrolledEventHandler — 此代理由事件的处理器使用。
- PageSelectedEventHandler — 此代理由 PageSelectedEvent 事件的处理器使用。

System.Windows.Controls.Primitives 名字空间是 System.Windows.Controls 名字空间的另一个子名字空间。如上所述，Primitives 子名字空间包括由其他更复杂的控件用作原语的控件。下述列表包括和 System.Windows.Controls.Primitives 名字空间关联的例子类。

- ButtonBase — 当在导出的类中覆盖时，定义相关的事件和属性，并提供相关输入事件的处理器。
- Popup — 创建包含内容的弹出窗口的控件。
- RangeBase — 表示有特定范围的元素的基类。这样的元素的例子包括滚动条和进度条 (progress bar)。这个类定义相关的事件和属性，并提供事件的处理器。
- RepeatButton — RepeatButton 控件在 Click 事件发生时添加重复语义。
- ScrollArea — ScrollArea 是用于滚动的有效元素。它包含剪辑的内容并提供属性来展现内容的偏移量和范围。它也提供缺省的输入处理，以使得可以用程序或通过键盘或鼠标滚轮来驱动滚动。
- ScrollBar — ScrollBar 类。
- Selector — 从它们的子元素中选取项的控件的基类。
- Slider — Slider 类。
- Thumb — 拇指控件支持滚动条的基本拖动功能和窗口的重设大小。

“IEnsureVisible”是和 System.Windows.Controls.Primitives 名字空间关联的例子接口。IEnsureVisible 在可视元素上实现以将子可视元素滚动/移动到视图中。

下面的列表包含和 System.Windows.Controls.Primitives 名字空间关联的例子枚举。

- ArrowButtonStates —
- CloseModeType — 描述弹出窗口如何响应各种鼠标事件。
- Part — Part 枚举用于表示构成滚动条的控件的语义用途。
- PartStates — ScrollBar 的部件状态。
- PlacementType — 描述应在屏幕上的何处放置弹出窗口。
- SizeBoxStates —

### 文档名字空间

文档名字空间 312 是用于创建具有丰富格式和丰富语义的文档的语义和格式元素的集合。在一个实施例中，“元素”是主要结合元素的类层次结构(称为“树”)使用的类。这些元素可以是交互的(如，通过键盘、鼠标或其他输入设备接收用户输入)，可以呈现图像或对象，并可以辅助安排其他元素。例子元素包括实现通用块的“Block”元素、表示包括表主体内容的“Body”元素、包含表内的表格数据的“Cell 元素”、表示包含在表头中的内容的“Header”元素，及用于将内容分成多个页面的“PageBreak”元素。

下面的列表包含由 System.Windows.Documents 名字空间展现的例子类。

- AdaptiveMetricsContext — AdaptiveMetricsContext 提供适应性流格式文档的根元素。一旦在 AdaptiveMetricsContext 元素中封装了子面板，则该面板的内容由 ReadingMetricsEngine(读取度量引擎 RME) 处理。子面板的大小被用来计算任何列的数量和大小及最佳字体大小和行高。
- Block — 实现不包括缺省呈现行为的通用块元素。
- BlockElement — 实现所有 Block 元素的基类。
- Body — 表示构成 Table 元素主体的内容。
- Bold — 实现从 Inline 导出的 Bold 元素。
- BreakRecord — 存储跨分页符继续格式化分页的内容所需的信息。从这个类继承以提供分页支持。该类为抽象类。
- Cell — Cell 包含 Table 内的表格数据。Cell 元素包含在 Row 中。
- CellCollection — 表格单元的有序集合。
- Column — Column 元素被用于分配 GridPanel 或 Table 的内容。
- ColumnCollection — ColumnCollection 是 Column 的有序集合。

- ColumnResult — 表示列的视图相关信息。
- ContainerParagraphResult — 提供对 Paragraph 对象的计算出的布局参数的访问， Paragraph 对象仅包含其他 Paragraph 对象。
- ContentPosition — 表示内容在段落中的位置。从这个类继承来描述相关内容的位置。这是一个抽象类。
- Document — Document 类的目的是分离文档内容和它周围的 UI “色彩”。 “分离” 指用户可以制作文档而无需考虑(且不会连累)它的 UI。 Document 类包含文档内容，通常是 TextPanel 或 FixedPanel 及其子元素。可视元素树(缺省为 PageViewer)通过 WPP 控件样式机制和此元素关联。
- DocumentPage — 表示和经受分页的文档的页关联的控件的布局信息。从此类继承进行实现来描述这些控件的布局信息。这是一个抽象类。
- DocumentPageParagraphResult — 提供对计算出的受分页影响的对象的布局参数的访问。
- FindEngine — 搜索算法的基类。
- FindEngineFactory — 搜索算法工厂。
- FixedPage — 提供对固定格式布局文档中的内容的单个页的访问。
- Footer — 表示构成 Table 元素页脚的内容。
- Header — 表示构成 Table 元素表头的内容。
- Heading — 实现将文本作为标题呈现的块级别元素。
- HyphenationDictionary — HyphenationDictionary 表示用于在应用中提供连字支持的字典。它可以包括内部字典和对外部字典的引用两者。内部字典有较高的优先级并在外部字典的条目之前应用。
- Hyphenator — Hyphenator 对象维护对 HyphenationDictionary 内的连字数据的引用同时也执行连字操作。
- Inline — 实现不招致任何缺省呈现行为的通用 Inline 元素。
- InlineElement — 实现作为所有内部元素基类的通用内部元素。
- Italic — 实现从 Inline 导出的 Italic 元素。
- LineBreak — 表示强制换行的标记元素。
- LineResult — 提供对计算出的一行文本行信息的访问。
- List — 实现 List 元素。 List 是设计为使用如黑点或编号这样的记号

格式化的块级别元素。

- ListElementItem — 实现 ListElementItem，它支持如黑点或编号这样的记号。
- Note — 实现 Note 元素，它类似于 HTML 中的注释元素。
- PageBreak — 表示用来将内容划分成各页的标记元素。
- PageDescriptor — 实现 PageDescriptor，它存储创建分页布局所需的信息。
- Paragraph — 实现用于在段落中呈现文本的块级别元素。呈现行为和 HTML 段落元素类似。
- ParagraphResult — 提供对计算出的 Paragraph 对象布局参数的访问。
- Row — 在 GridPanel 或 Table 元素内定义行。
- RowCollection — RowCollection 表示 Row 的有序集合。
- RowGroup — 指定 Table 或 GridPanel 中一组行的缺省属性。
- Section — 实现通用容件元素。呈现行为和 HTML 中的 DIV 元素类似。
- SmallCaps — 实现内部 SmallCaps 元素。SmallCaps 是作为字符的小体大写字母版本呈现以用于(如在标题内)强调的印刷体。
- Subscript — 表示内部 Subscript 元素。Subscript 字符被写在在其他字符的正下方、左下方、或右下方。
- Superscript — 表示内部 Superscript 元素。Superscript 字符通常 是字母或数字并在其他字符的正上方、左上方、或右上方呈现。
- Table — Table 被用于使用标记语言(如，“XAML”)以表格形式显示复杂的数据。
- TextArray — 用于文本访问和处理的基础 API。
- TextChangedEventArgs — TextChangedEventArgs 定义当 TextArray 改变时发送的事件参数。
- TextElement — TextElement 对 TextTree 提供 TextRange 工具。它是不可变的、连续的有固定端点的 TextRange。它提供 ContentElement 输入、聚焦点和事件支持。它还提供 DependencyObject 属性支持。
- TextNavigator — 这可以枚举文本内容。实现可移动的 TextPosition。它可以根据运行的或定位在文本中已知位置的文本移动。

- `TextParagraphResult` — 提供对计算出的文本布局参数的访问，这些参数包括浮动的对象和图形。
- `TextPosition` — 这是表示 `TextArray` 中特定位置的对象。表示文本中位置的紧凑对象在文本改变时自动地保持位置。比较操作仅适用于同一 `TextArray`(同一上下文)内的位置。`TextPosition` 可以是静态的或可移动的。`IsChangeable` 属性区分位置的类型。
- `TextRange` — `TextRange` 是向零或多个子范围的通用关联提供属性的抽象类。在导出类中定义子范围处理。
- `TextRangeMovable` — `TextRangeMovable` 是可移动 `TextRange` 的抽象类。它提供基于 `TextUnit` 移动起始和终止端点的能力。
- `TextTreeChangedEventArgs` — `TextChangedEventArgs` 定义在 `TextArray` 改变时发送的 事件参数。
- `TextTreeDumper` — `TreeDumper` 是树检验类，由于打包问题它是一个公共类。
- `TextTreeNavigator` — 这是表示 `TextTree` 中特定的可移动位置的对象。它是 `TextNavigator` 的具体实现，并只用在 `TextTree` 中。
- `TextTreePosition` — 这是表示 `TextTree` 中特定的不可变的位置的对象。它是 `TextNavigator` 的具体实现，并只用在 `TextTree` 中。
- `TextTreeRange` — 对 `TextTree` 提供 `TextRange` 工具。它是用可移动端点的可改变的、连续 `TextRane`。
- `TextTreeRangeContentEnumerator` — 直接在 `TextTreeRange` 下面的直接子对象的枚举器。
- `TextUnit` — 文本导航的可扩展单位。
- `TextUnits` — 对 `TextPosition` 和 `TextRange` 通常使用的文本单位。
- `Typography` — 提供对 OpenType 字体属性的丰富集合的访问。
- `UIElementParagraphResult` — 完全由 `UIElement` 构成的段落的 `ParagraphResult`。用于浮动的对象、图形和嵌入的块级别 `UIElement`。
- `Underline` — 实现从 `InlineElement` 导出的 `Underline` 元素。

下面的列表包含和 `System.Windows.Documents` 名字空间关联的例子接口。

- **IDocumentContentHost** — 在内容宿主机上实现此接口以使得该宿主机的子对象可以在内容改变时通知该宿主机。
- **IDocumentFormatter** — 在元素上实现此接口以提供对如分页这样的文档特性的支持。
- **ITextDocumentResult** — 实现此接口来保持文档的列信息。
- **ITextParagraphResult** — 实现此接口以提供文本和对文本段落的定位信息。

下面的列表包含和 `System.Windows.Documents` 名字空间关联的例子枚举。

- **ElementEdge** — 这标识 `TextPosition` 被定位的对象的边缘。
- **FindAdvancedOptions** — 由 `FindAlgorithm`( 搜索 初始化 ) 和 `TextRangeMovable/TextSelection`( 简化的搜索执行 ) 类使用的高级搜索选项。
- **FindOptions** — 由 `TextBox.Find` 方法使用的简化搜索选项。
- **LogicalDirection** — `LogicalDirection` 定义在文本中移动的逻辑方向。它也用于确定在 `TextPosition` 插入内容时将 `TextPosition` 移至何处。
- **TextArrayRunType** — 识别 `TextPosition` 所处的范围，并考虑 `LogicalDirection`。
- **TextChangeOptions** — 对 `CanChangeText` 的可能的文本改变。
- **TextMoveOptions** — 这通过指定终止导航的条件控制 `TextNavigator` 的移动。

下面的列表包含和 `System.Windows.Documents` 名字空间关联的例子代理。

- **ObjectCloneDelegate** — 在复制或移动 `TextArray` 的一部分时提供 `DependencyObject` 的克隆或拷贝的回调用方法。
- **TextChangedEventHandler** — 在每次增加内容到 `TextTree` 或从 `TextTree` 移除内容时用 `TextChangedEventArgs` 调用 `TextChangedEventHandler` 代理。

## 形状(shape)名字空间

形状名字空间 314 是用于创建图像和对象的矢量图形元素的集合。使用矢量图形元素允许简单地重新设置元素的大小以适合特殊接口或显示设备的要求。例子元素包括画椭圆的“Ellipse”元素、在两点之间画直线的“Line”元素、画矩形的“Rectangle”元素，以及将多边形作为构成封闭形状的连接的直线序列画出的“Polygon”元素。

下面的列表包含由 System.Windows.Shapes 名字空间展现的例子类。

- Ellipse — 画椭圆。
- Glyphs — 表示如“XAML”这样的标记语言中的字形。Glyphs 用于表示字体。
- Line — 画两点之间的直线。
- Path — 画一系列的相连直线和曲线。
- Polygon — 画多边形(构成封闭形状的连接的直线序列)。
- Polyline — 画一系列的连接直线。
- Rectangle — 画矩形。
- Shape — 对形状元素(如椭圆、多边形和矩形)提供基础功能的抽象类。

## 数据名字空间

数据名字空间 316 包括用于绑定属性元素到数据源、数据源类及数据集合与视图的数据特定的实现的类和接口。这些类和接口也用于处理数据条目中的例外并允许基于各种数据源中的信息在运行时创建用户接口。可以用文本形式显示数据或用数据来改变显示的格式，例如，若美元数量为负数则以红色显示。例子类包括“Bind”类，它表示管理动态属性用户接口和源数据之间的绑定的绑定说明对象，及“XmlDataSource”类，它充当用于绑定到 XML 内容节点的数据的数据源。

面向对象的应用通常用定义一段数据的值以及可以对该数据执行的操作的类来表示数据。术语“数据项”指一个这样的对象。应用软件可以处理单独的数据项或数据项的集合。它们可以用三种方式使用数据项：(a) 将来自外部数据源(如文件系统、远程服务器、数据库等等)的数据转换为对应的存储器内的数据项，并将修改后的数据项转换回到这些数据源所期望的形式；(b)

使用数据为中心及应用为中心的逻辑的组合操作数据项；(c) 通过用户接口向用户呈现由数据项实现的数据。数据名字空间 316 对这些任务中的第一个和第三个提供支持。

第一个任务，获取来自外部数据源的数据，由“数据源”对象支持。数据源对象通常被定义为页范围或应用范围的资源，并充当数据的信关。数据源实现 `IDataSource` 接口，此接口定义在数据名字空间中的类获取对数据的访问的标准机制。特殊的数据源对象实现通过使用适合于特殊源的机制检索实际数据的逻辑。在一个实施例中，数据名字空间包括四个数据源类：

1. `XmlDataSource`, 用于检索表示为 XML 的数据。
2. `SqlDataSource`, 用于从 SQL 数据, 如 Microsoft SQL Server, 库检索数据。
3. `WinFSDDataSource`, 用于从 WinFS 服务检索数据。
4. `ObjectDataSource`, 用于从应用软件定义的任意对象检索数据。

应用软件也可以定义裁剪成专用目标源的它们自己的数据源类。

数据源类负责从外部源检索数据并将它转换为适合于由绑定类使用的一个或多个数据项。如果需要数据项的集合，则应用软件可以使用任何来自 .Net 框架的标准集合类，如 `Array`、`ArrayList`、`Hashtable` 等等、任何来自 `System.Data` 名字空间的数据为中心的集合类，如 `Dataset`，或来自数据名字空间数据为中心的集合类，如 `ArrayListDataCollection`。后面的这些类支持改变通知，即，当应用软件通过增加项、移除项、对集合排序等等改变集合时，集合发送通知。绑定类侦听这些通知并自动地更新用户接口来反映此改变。

一旦数据已被转换为存储器内的数据项，应用软件就可以使用这些项执行计算并可以修改这些项将其作为计算的结果。使用数据为中心的操作（由数据项类定义）和应用为中心的操作（由应用软件自身定义）的组合来执行这些动作。这些动作可以由应用软件自动触发，或响应用户的动作触发。来自数据名字空间的特殊支持或合作不是必须的，从而在应用内提供逻辑和表示的清楚的划分。

第三个有关数据的任务，通过用户接口呈现数据，由数据名字空间的“绑定”类支持。这些类使得应用软件能够描述数据项属性（源）和用户接口属性（目标）之间的对应关系（绑定）。术语数据绑定（或简单地，绑定）指建立这样的对

应关系。例如，应用软件可以选择将 `Textbox` 控件的 `Text` 属性和数据项的 `CustomerName` 属性进行数据绑定。这样之后，控件将自动显示客户的名字，在应用改变数据项的时候更新显示，并在用户输入新名字到控件中时更新数据项。

使用 `Bind` 类来描述这类对应关系，并使用 `Binding` 类来实现这类对应。任何数量的 UI 属性可以共享同一描述 (`Bind`)，但是每个属性有它自身唯一的 `Binding`，`Binding` 包含该特殊实例的状态。该描述包括下面有关所需对应关系的信息：

- `Path` — 用作绑定的源的数据项属性的名称。这可以是简单的属性名称，或包括子对象和下标(如，当源属性有复杂类型的值时)的更加复杂的表达式，如“`ShippingAddress.Line[2]`”。当数据源为 XML 时，路径为 XPath 表达式。
- `BindType` — 对应关系是否为单向、双向或一次性的。在单向绑定中，对数据项的改变导致对用户接口属性的更新；数据率单向流动—从源到目标。在双向绑定中，数据双向流动；除了单向行为以外，对用户接口属性的改变导致对数据项属性的更新。在一次性绑定中，数据项属性被用来初始化用户接口属性，但是改变将不在任何方向上扩散。
- `Source` — 在何处获取源数据项的描述。这可以来自数据源对象、来自某些其他用户接口元素，或来自目标元素的 `DataContext` 属性的值。
- `UpdateType` — 在双向绑定中何时更新源属性：`Immediate`、`OnLostFocus` 或 `Explicit` 之一。`Immediate` 更新在用户接口属性一改变时立即发生。`OnLostFocus` 更新被延迟到目标元素失去键盘焦点的时候—这适合于 `TextBox` 控件，以避免在每次按键之后就更新的开销。`Explicit` 更新在应用软件明确调用它们时发生。
- `Transformer` — 实现 `IDataTransformer` 接口的对象。这向应用软件给出在用户接口中使用之前修改数据项值的方法。修改可以是简单的类型转换(如，在 `Background` 属性到 `BalancedOwed` 数据属性的绑定中，应用软件可以转换负的余额为红色背景而正的余额为绿色背景)，或特定于应用的转换(如，在 `Text` 属性到 `NetWorth` 数据属性的绑定中，如果 `NetWorth` 超过一百万美元则应用可以显示“富有”，如果 `NetWorth` 在十万美元到一百万美元之间则显示“中产阶级”，而如果 `NetWorth` 小

于十万美元则显示“贫穷”。Transformer 是有助于根据数据分离表示的简单但强大的工具。

除了一次性绑定之外的所有绑定都依赖于在数据属性改变时得到通知，从而可以对用户接口做出相应的改变。绑定类将 IPropertyChange 接口(来自 System.ComponentModel 名字空间)作为一种实现所需通知的方法。

下面的表列出了由 System.Windows.Data 名字空间展现的成员。

类	
ArrayListCollectionView	封装对 ArrayListDataCollection 集合类的集合视图支持。不能从这个类继承。
ArrayListDataCollection	向数组列表数据集合的内置实现提供底层的集合视图接口。它也实现 ICollectionChange 以在增加项、移除项或刷新整个集合时提供通知。
Bind	表示绑定说明对象，用来管理动态属性用户接口 (UI) 和源数据之间的绑定。
Binding	提供对绑定的单个运行时实例的访问。不能从这个类继承。
BindingListView	用于 Microsoft® ActiveX® Data Objects (ADO) 数据视图的集合视图类。
CollectionContainer	这个类的对象包含现有的集合结构—例如，ArrayListDataCollection 或 ItemCollection 内的某些其他 DataSet。
ContextAffinityCollectionView	实现包括对上下文关联性的检验的集合视图。
DataContextObjectRef	支持对用作绑定的数据上下文的对象的对象引用。不能从这个类继承。
DataSourceObjectRef	支持对数据源的对象引用。不能从这个类继承。
DataTransferEventArgs	封装数据传输事件的参数。这些事件是具体地基于 DataTransferEventHandler

	代理由指定的处理程序处理的所发送的事件。
ElementObjectRef	表示对元素的对象引用，该对象由其元素 ID 指定。不能从这个类继承。
ExplicitObjectRef	表示对元素的明确的对象引用。不能从这个类继承。
ListCollectionView	基于 IList 实现对各集合的一集合视图。
ObjectDataSource	充当数据绑定的数据源。可绑定的数据项可被指定为通用语言运行库类型。
ObjectRef	用 作 ElementObjectRef 、 ExplicitObjectRef 和 TypeObjectRef 的父类的抽象类。
ParameterCollection	这个类的对象包含 SqlDataSource 的命名参数(和它们相应的值)的集合。
QueryCommand	这个类表示将向数据库提交的单个 select 语句。
RefreshCompletedEventArgs	封装 在 ObjectDataSource 的 RefreshCompleted 事件或 XmlDataSource 的 RefreshCompleted 事件中传递的参数。
SqlCommandList	SQL 命令的列表及用它们来填充的表名。
SqlDataSource	SqlDataSource 从 Microsoft SQL Server 获取数据用于数据绑定。
TransformerSource	允许资源引用定义为当前应用背后的代码的转换器类。
TypeObjectRef	支持通过类型的对象引用。不能从这个类继承。
WinFSDatasource	WinFSDatasource 有助于存储在 WinFS 中的数据和 Avalon 应用的数据组定。
XmldatamespaceManager	用于说明在 Xml 数据绑定 XPath 查询中

	使 用 的 名 字 空 间 的 。 XmlDataNamespaceManager 类。
XmlDataSource	充当绑定到可扩展标记语言 (XML) 内容节点的数据的数据源。
XmlNamespace	说明 XML 数据源内单独的名字空间。

接口	
IContains	用于创建说明集合视图过滤准则的类。
IDataSource	支持数据源对象的创建。数据源对象被用于用于数据绑定的数据的通用表示。
IDataTransformer	提供使能在客户机方转换绑定数据的方法。

枚举	
BindFlags	描述绑定的特殊属性。参见使用“Longhorn”标记语言(代码名称为“XAML”)的绑定说明的用法。参见用于指定绑定类型(单向、双向等等)的枚举的 BindType。
BindStatus	绑定的状态。
BindType	描述数据值的改变如何来自绑定的源属性和目标属性并传输给它们。
SqlDataSourceMode	枚举 SqlDataSource 可能具有的模式。模式确定当应用软件检索来自 Data 属性的值时返回何种类型的数据。
UpdateType	指定在绑定中应何时发生对数据源的更新(目标到源的数据传输)。设置这些值仅在绑定的 BindType 被设置为 TwoWay(或保持为缺省值)时相关。

代理	
DataChangedEventHandler	表示处理由实现 <code>IDataSource</code> 的数据源触发的 <code>DataChanged</code> 事件的方法。
DataTransferEventHandler	表示处理由 <code>Binding</code> 触发的数据传输事件的方法。
RefreshCompletedEventHandler	表示处理 <code>ObjectDataSource.RefreshCompleted</code> 和 <code>XmlDataSource.RefreshCompleted</code> 事件的方法。

### 媒体名字空间

媒体名字空间 318 提供各种媒体类。应用软件开发者及组件开发者可以使用这些类来开发各种表示功能。媒体名字空间 318 中的例子类包括支持特定图像效果(如，模糊和灰度)的“`ImageEffect`”类和提供使用单色、渐变色、图像、视频等等填充区域的机制的“`Brush`”类。

媒体名字空间 318 包括子名字空间 `System.Windows.Media.Animation`，该子名字空间包括允许开发者用动画呈现属性并协调带一组时间线的动画集合。动画是在一段时间上改变值的对象。动画效果包括在显示上移动一对象，及改变一对象的大小、形状或颜色。提供多个动画类以实现各种动画效果。效果可以通过关联动画和元素的属性值来实现。例如，为了创建淡入和淡出视图的矩形，一个或多个动画和矩形的透明度属性关联。

媒体名字空间 318 也包括提供各种文本服务的子名字空间 `System.Windows.Media.TextFormatting`。例如，“`TextFormatter`”文本引擎提供划分文本行并格式化在显示上呈现的文本的服务。“`TextFormatter`”能够处理不同的文本字符格式和段落样式，且能够处理国际化文本布局。

下面的表列出由 `System.Windows.Media` 名字空间展现的例子成员。

类	
<code>ArcSegment</code>	表示两点之间的椭圆弧。
<code>AudioData</code>	根据时间节点的状态使能播放音频文件。

AudioDataConverter	AudioDataConverter
BezierSegment	表示在两点之间画出的立方贝塞尔曲线。
Brush	提供使用单色 (SolidColorBrush)、渐变色 (LinearGradientBrush)、RadialGradientBrush)、图像 (ImageBrush)、视频等等填充区域的通用方法。
BrushConverter	用于在 Brush 对象和其他对象类型之间进行转换。
Brushes	实现一组预定的单色。
CloseSegment	表示连接 PathFigure 的最后一个点和其起始点的线。
CodecFilter	枚举编码解码器的过滤器。只枚举那些和属性匹配的编码解码器。
CodecInfo	有关特定编码解码器和创建该编码解码器的工厂的信息。这是从编码解码器枚举器返回。
ColorCollection	
ColorCollectionConverter	ColorCollectionConverter — 用于在其他类型的实例和 ColorCollection 实例之间进行转换的转换器类。
ColorContext	
ColorConverter	用于在 Color 对象和其他对象类型之间进行转换。
Colors	实现一组预定的颜色。
ContainerVisual	管理 Visual 对象的集合。
DashArrays	DashArrays — DashArrays 类是静态的，并包含用于众所周知的虚线样式的属性。

DoubleCollection	
DoubleCollectionConverter	DoubleCollectionConverter — 用于在其他类型的实例和 DoubleCollection 实例之间进行转换的转换器类。
Drawing	Drawing 为 2D 绘画原语的列表。
DrawingBrush	DrawingBrush — 此 TileBrush 定义其内容为 Drawing。
DrawingContext	绘画上下文。
DrawingVisual	包含被画出的图像内容的可视组件。
EllipseGeometry	表示圆或椭圆的几何。
FontFamily	字体族。
FormattedText	FormattedText 类是 Avalon MIL 简单文本 API 的部分，它针对需要增加某些简单文本到 MIL 可视组件的程序员。
Geometry	对所有的几何图形类（如 EllipseGeometry、RectangleGeometry 和 PathGeometry）提供基础功能的抽象类。对象的 Geometry 类可以用于剪切、点击测试和呈现 2D 图形数据。
GeometryCollection	表示 Geometry 对象的集合。
GetPageEventArgs	GetPageEventArgs 类。
GlyphRun	GlyphRun 类。
GlyphTypeface	对应于盘上的字体文件的物理字体。
GradientBrush	描述渐变色填充的抽象类。从 GradientBrush 导出的类描述解释渐变色停止点的不同方法。
GradientStop	描述渐变色中转换点的位置和颜色。
GradientStopCollection	表示 GradientStop 渐变色停止点的集合。
HitTestParameters	这是包装用于点击测试的诸参数的基础

	类。
HitTestResult	这个基础类返回在点击测试中点击中的可视组件。
HwndInterop	HwndInterop。
HwndVisual	
HyphenationCandidate	描述一个连字候选。
ICCPProfile	
ImageBrush	用图像填充区域。此类可用于将图像指定作为其他对象的填充和背景。
ImageCodecCollection	系统上编码解码器的集合(实际的 CodecInfo)。
ImageCodecEnumerator	图像帧的枚举器。
ImageColorTransform	ImageColorTransform 在图像管线上执行颜色管理。
ImageData	包含图像和相关数据。
ImageDataBuilder	此对象被用来构建 ImageData 对象。
ImageDecoder	ImageDecoder 是图像帧的组件。每个图像帧都是 ImageSource。不像 ImageSource, ImageDecoder 不是不可变的对象且可以重新初始化为不同的图像流。然而, 任何返回的 ImageSource(帧)都是不可变的。
ImageDecoderBmp	内置的 Microsoft Bmp(位图)解码器。
ImageDecoderGif	内置的 Microsoft GIF 解码器。
ImageDecoderIcon	内置的 Microsoft 图标解码器。
ImageDecoderInternal	仅在内部使用。
ImageDecoderJpeg	内置的 Microsoft Jpeg 解码器。
ImageDecoderPng	内置的 Microsoft Png 解码器。
ImageDecoderTiff	内置的 Microsoft Tiff 解码器。
ImageEffect	ImageEffect 类是所有图像效果(模糊、

	灰度等等)的基类。效果可能没有任何输入但效果应至少有一个输出。缺省的实现假设是这样。如果要用输出/多个输出播放导出的效果，则应确保至少有一个输出。
ImageEffectBlur	高斯模糊效果。它是单输入、单输出效果。警告：如果缩放此效果(即，Input.ScaleX 和 Input.ScaleY 不是 1)且 Expand 是真，则可能输出比 PixelWidth 和 PixelHeight 更大或更小的维度。调整用于复制的象素缓冲来避免问题。
ImageEffectFlipRotate	此效果可以在 X 或 Y 方向翻转图像及将图像旋转 90 度的倍数。
ImageEffectGammaCorrect	此效果改变图像的灰度系数。
ImageEffectGlow	执行发光效果。它是单输入、单输出效果。
ImageEffectGrayscale	转换图像为灰度。它是单输入、单输出效果。
ImageEffectNegate	对图像求逆。它是单输入、单输出效果。
ImageEffectSharpen	模糊遮蔽罩(mask)。它是单输入、单输出效果。
ImageEffectSource	ImageEffectSource 类的实现。
ImageEffectSourceCollection	图像效果输出的集合。
ImageEffectTint	色调构造器。它是单输入、单输出效果。
ImageEncoder	ImageEncoder 收集一组(ImageSource 的)帧及它们相关的缩略图和元数据并将它们保存到指定的流。除了特定于帧的缩略图和元数据，如果编码解码器支持的话，也可以有图像范围(全局)的缩略图和元数据。

ImageEncoderBmp	内置的 Bmp 文件编码器。
ImageEncoderGif	内置的 Gif 文件编码器。
ImageEncoderInternal	ImageEncoderInternal 收集一组 (ImageSource 的) 帧及它们相关的缩略图和元数据并将它们保存到指定的流。除了特定于帧的缩略图和元数据，如果编码解码器支持的话，也可以有图像范围(全局)的缩略图和元数据。
ImageEncoderJpeg	内置的 Jpeg 文件编码器。
ImageEncoderPng	内置的 Png 文件编码器。
ImageEncoderTiff	内置的 Tiff 文件编码器。
ImageExchangeMetaData	ImageExchangeMetaData 这个类被用于访问和设置具有 Exif 样式元数据的 ImageFile 的元数据。MetaData 作为键 / 值的配对来存储，其中键不一定是唯一的。这个类提供对图像内所有元数据的通用访问，并对某些众所周知的属性展现 CLR 属性。
ImageExchangeProperty	ImageExchangeProperty — ImageExchangeID 和作为该属性值的对象的元组。
ImageMetaData	ImageMetaData 这个类用于访问和设置 Image 的元数据。此类也展现 CodecMetaData 属性，用它展现访问此图像源数据的专用于编码解码器的方法。
ImagePalette	ImagePalette 类。
ImageSizeOptions	图像的大小设置选项。将基于这些选项缩放得到的图像。
ImageSource	定义用于图像管线的方法、属性和事件，包括解码器和效果。

ImageSourceCollection	系统上的编码解码器的(实际上是 ImageSource 的)集合。
ImageSourceConverter	ImageSourceConverter
IntegerCollection	
IntegerCollectionConverter	IntegerCollectionConverter — 在其他类型的实例和 IntegerCollection 实例之间进行转换的转换器类。
LinearGradientBrush	定义用于填充区域的线性渐变色。
LineGeometry	表示直线的几何图形
LineSegment	表示两点之间的线段。不像 LineGeometry 对象， LineSegment 应包含在 PathFigure 内。
MatrixTransform	创建任意的用于处理二维平面上的对象或坐标系统的仿射矩阵变换。
MediaData	MediaData。用于回放音频/视频内容。
MediaSystem	MediaSystem 类控制媒体层。
NineGridBrush	用图像填充整个区域。拉伸图像以拟合则所定义的边界。
PathFigure	表示几何图形的分段，二维几何图形线段的单个连接序列。
PathFigureCollection	
PathFigureConverter	PathFigureConverter
PathGeometry	表示可以由弧、曲线、椭圆、直线和矩形组成的复杂形状。
PathGeometryConverter	PathGeometryConverter
PathSegment	表示 PathFigure 对象的一段的抽象类。从 PathSegment 导出的类，如 ArcSegment 、 BezierSegment 和 LineSegment，表示特定类型的几何图形线段。

PathSegmentCollection	表示 PathSegment 对象的列表。
PathSegmentConverter	PathSegmentConverter
Pen	描述如何画出形状的轮廓。
PixelFormats	PixelFormat — 所支持的像素格式的集合。
PointCollection	
PointCollectionConverter	PointCollectionConverter — 用于在其他类型的实例和 PointCollection 实例之间进行转换的转换器类。
PointHitTestParameters	这是用于指定用一点来作点击测试的参数的类。
PointHitTestResult	此类返回该点和在点击测试中点击到的点和可视组件。
PolyBezierSegment	PolyBezierSegment
PolyLineSegment	PolyLineSegment
PolyQuadraticBezierSegment	PolyQuadraticBezierSegment
PrintContext	PrintContext 包含打印机交互的状态和上下文。
QuadraticBezierSegment	QuadraticBezierSegment
RadialGradientBrush	定义用于填充对象的放射渐变色。焦点定义渐变色的开始，圆定义渐变色的终点。
RectangleGeometry	表示矩形的几何形状。
RetainedVisual	RetainedVisual
RotateTransform	用于在二维 x-y 平面上围绕指定的点旋转一对象。
ScaleTransform	在二维 x-y 平面上从定义的中心点开始来缩放一对象。在 x 和 y 方向定义从这个中心点缩放因子。
SkewTransform	表示二维扭斜。

SolidColorBrush	表示单色的均匀填充。
StartSegment	StartSegment。
SubLineCollection	子线段的集合。子线段可以是 GlyphRun、LineOver、Inine 对象这些类型中一种的对象。
TileBrush	描述用一个或多个“贴图”填充区域的方法的抽象类。导出的类定义可以使用的不同类型的贴图；例如，ImageBrush 使用户可以用图像填充区域。
Transform	用作二维平面上所有类型的转换，包括旋转 (RotateTransform)、缩放 (ScaleTransform)、扭曲 (SkewTransform) 和平移 (TranslateTransform) 的父类的抽象类。这个类层次结构不同于 Matrix 结构，因为它是一个类，还因为它支持动画和枚举语义。
TransformCollection	用于创建和处理 Transform 对象的一个列表。
TransformConverter	用于在 Transform 对象和其他对象类型之间进行转换。
TranslateTransform	在二维 x-y 平面上平移一对象。
Typeface	Typeface 为字体、加浓、样式和伸展的组合。
VectorCollection	
VectorCollectionConverter	VectorCollectionConverter — 用于 在其他类型的实例和 VectorCollection 实例之间进行转换的转换类。
VideoData	使能根据时间节点的状态播放视频文件。

VideoDataConverter	VideoDataConverter
Visual	所有可视组件类型的基类。它提供所有的可视组件共有的服务和属性，包括点击测试、坐标转换和边界框计算。
VisualCollection	Visual 对象的有序集合。
VisualManager	呈现 Visual 对象树到呈现目标，呈现目标通常是一窗口。

接口	
IHyphenate	IHyphenate 是连字服务提供器的接口。
IRetainedRender	如果此接口在从 RetainedVisual 导出的类上实现，则以校验模式的 RetainedVisual 操作，即，图形子系统将以缓慢的方式调用 OnRender。(如，如果 Visual 第一次在屏幕上出现)。注意 OnRender 可以由系统在任何时间调用。
IVisual	此接口定义从 Visual 对象可得到的共同的方法和服务。

枚举	
BrushMappingMode	BrushMappingMode — 描述是否应将某些值看作绝对的局部坐标或它们是否应作为边界框大小倍数的枚举。
ChannelDescription	描述像素数据的每个通道的顺序。
ColorInterpolationMode	ColorInterpolationMode — 这确定渐变中的颜色如何内插变化。
CombineMode	指定用来组合两个几何图形区域的方法。
FillRule	

GradientSpreadMethod	指定在指定的渐变向量或空间之外如何画出渐变色。
HitTestFilterBehavior	在进行点击测试时过滤可视组件的行为。
HitTestResultBehavior	枚举当在点击测试中出现有效点击时的控件行为。
HorizontalAlignment	HorizontalAlignment 枚举被用于描述内容如何在容件内水平地放置。
HyphenationRule	所支持的连字规则。
ImagePaletteType	预定义的调色板类型。
MediaState	包含媒体的当前状态。
PenDashCap	PenDashCap — 描述在虚线内虚线终端的画法的枚举。
PenLineCap	描述在直线或线段终端的形状。
PenLineJoin	PenLineJoin — 描述直线转角的画法的枚举。
Rotation	所应用的旋转；仅支持 90 度的倍数。
StandardColorSpace	
Stretch	Stretch — 描述源矩形应如何拉伸以拟合到目标矩形的枚举。
StyleSimulations	字体样式模拟。
TiffCompressOptions	用于保存 TIFF 图像的压缩选项
TileMode	TileMode — 描述直线终端画法的枚举。(译者注：疑意思不对)
VerticalAlignment	VerticalAlignment 枚举用于描述如何在容体内垂直地放置内容。

结构	
CharacterIndexer	此类是实现字符的命名索引的辅助器。
Color	以 alpha、红、绿和蓝通道表示颜色。

GlyphIndexer	此类是实现字形度量的命名索引的辅助器。
ImageExchangeID	ImageExchangeID — 这个类是可以在 ImageMetaData 实例中用作属性的键值的类型。
ImageExchangeMetaDataEnumerator	ImageExchangeMetaDataEnumerator — ImageExchangeMetaData 的枚举器。包含 IEnumarator 接口及 API 的强类型版本。
ImageFrameEnumerator	Image 帧的枚举器。
ImageMetaDataRational	ImageMetaDataRational 类 表示 为 有符号的分子和有符号的分母。有理数的有效值是分子/分母。
ImageMetaDataUnsignedRational	有理数类表示为无符号的分子和无符号的分母。有理数的有效值是分子/分母。
ImagePaletteColor	ImagePaletteColor 结构。
IntegerRect	由整数值组成的矩形。通常用来从图像中(以象素)指定关注的源矩形。
Matrix	表示用于在二维空间中的转换的 $3 \times 3$ 的矩阵。由于“Avalon”只允许仿射转换，Matrix 结构包含六个条目，而非九个。
NamedStringIndexer	这个类为实现在多种文化中本地化的字符串的命名索引的辅助器。
PixelFormat	对图像和基于象素的表面的象素格式定义。

代理	
GetPageEventHandler	代理 GetPageEventHandler。
HitTestFilterDelegate	代理点击测试器来控制是否测试可视组

	件的子元素。
HitTestResultDelegate	代理点击测试器来控制可视组件上的点击信息的返回。

下面的表列出由 System.Windows.Media.Animation 名字空间展现的例子成员。

类	
Animatable	任何并非从 DependencyObject 导出但是有可以用动画呈现的属性的类应从这个类导出。
AnimationCollection	这个抽象类为动画集合，如 ColorAnimationCollection 、 DoubleAnimationCollection 和 SizeAnimationCollection 提供基础功能。
AnimationEffect	覆盖这个类来实现可以参与呈现处理以在呈现时在多个元素上实例化动画的元素级别动画。
AnimationEffectCollection	包含 AnimationEffect 的集合。
BoolAnimationCollection	表示 BoolModifier 动画的集合。
BoolModifier	
BoolTimedModifier	
ByteAnimationCollection	表示 BoolModifier 动画的集合。
ByteModifier	
ByteTimedModifier	
CharAnimationCollection	表示 CharModifier 动画的集合。
CharModifier	
CharTimedModifier	
ColorAnimation	用动画呈现属性的颜色值。
ColorAnimationCollection	表示 ColorModifier 动画的集合。

ColorKeyFrameCollection	
ColorModifier	
ColorTimedModifier	
DecimalAnimationCollection	表示 DecimalModifier 动画的集合。
DecimalModifier	
DecimalTimedModifier	
DoubleAnimation	用于对接受 Double 值的属性进行动画呈现。
DoubleAnimationCollection	表示 DoubleModifier 动画的集合。
DoubleKeyFrameCollection	
DoubleModifier	
DoubleTimedModifier	
FLOATAnimation	用于对接受 Single 值的属性进行动画呈现。
FLOATAnimationCollection	表示 FloatModifier 动画的集合。
FLOATKeyFrameCollection	
FLOATModifier	
FLOATTimedModifier	
INTAnimationCollection	表示 IntModifier 动画的集合。
INTModifier	
INTTimedModifier	
LENGTHAnimation	用于对接受 Length 值的属性进行动画呈现。
LENGTHAnimationCollection	表示 LengthModifier 动画的集合。
LENGTHKeyFrameCollection	
LENGTHModifier	
LENGTHTimedModifier	
LONGAnimationCollection	表示 LongModifier 动画的集合。
LONGModifier	
LONGTimedModifier	

MatrixAnimationCollection	表示 MatrixModifier 动画的集合。
MatrixModifier	
MatrixTimedModifier	
Modifier	
ObjectAnimationCollection	表示 ObjectModifier 动画的集合。
ObjectModifier	
ObjectTimedModifier	
PathAnimation	可以在 MatrixAnimationCollection 内部使用这个动画以沿着一路径移动可视组件对象。
PointAnimation	用于对接受 Point 值的属性进行动画呈现。
PointAnimationCollection	表示 PointModifier 动画的集合。
PointKeyFrameCollection	
PointModifier	
PointTimedModifier	
RectAnimation	用于对接受 Rect 值的属性进行动画呈现。
RectAnimationCollection	表示 RectModifier 动画的集合。
RectKeyFrameCollection	
RectModifier	
RectTimedModifier	
ShortAnimationCollection	表示 ShortModifier 动画的集合。
ShortModifier	
ShortTimedModifier	
SizeAnimation	定义基于一对象的 Size 的动画。通过提供 Size 信息，对象可以在一段时间内显得缩小或变大。
SizeAnimationCollection	表示 SizeModifier 动画的集合。
SizeKeyFrameCollection	

SizeModifier	
SizeTimedModifier	
StringAnimationCollection	表示 StringModifier 动画的集合。
StringModifier	
StringTimedModifier	
Timeline	对定时对象保持运行时定时状态。
TimelineBuilder	可用于创建 Timeline 对象的对象。
TimeManager	控制整个定时树的对象。
TimeSyncValueTypeConverter	执行涉及 TimeSyncValue 值的类型转换的对象。
TimeTypeConverter	执行涉及 Time 值的类型转换的对象。
VectorAnimation	用于对接受 Vector 值的属性进行动画呈现。
VectorAnimationCollection	表示 VectorModifier 动画的集合。
VectorKeyFrameCollection	
VectorModifier	
VectorTimedModifier	

接口	
IClock	表示所提供线性时间值的对象。
IModifier	定义修改器对象的基本行为。修改器是取称为基本值的一个特定类型的对象，并返回同一类型的另一个对象作为其输出的对象。
ITimingControl	定义时间线和定时对象的行为。
ITimingControlBuilder	表示可以构建时间模板的对象。

枚举	
AnimationType	描述动画的行为。
CloneType	CloneCore 可以请求的克隆类型。

InterpolationMethod	描述动画如何计算其输出值。
KeyTimeType	不同类型的 KeyTime。
TimeEndSync	endSync 属性的值，它指定容件如何基于子元素的持续时间计算其简单持续时间。
TimeFill	指定时间线在它不再有效时如何表现。
TimeRestart	Timeline. Restart 属性的值。
TimeSeekOrigin	指示时间线的位置；用于通过定义偏移量所应用的位置指定 ITimeControl 接口的 Seek 方法的行为。
TimeSyncBase	同步开始或终止值的事件。

结构	
ColorKeyFrame	
DoubleKeyFrame	
FloatKeyFrame	
KeySpline	这个类用于传递关键样条的数组到动画片段的 KeySplines 属性。
KeyTime	KeyTime 用于指定 KeyFrame 相对于动画的时间何时出现。
LengthKeyFrame	
PointKeyFrame	
RectKeyFrame	
SizeKeyFrame	
Time	表示时间的值，与时间算术操作关联。
TimelineEnumerator	枚举 TimelineList 集合中的项。
TimeSyncValue	表示时间线的绝对或相对开始或终止时间的值。
VectorKeyFrame	

下面的表列出由 System.Windows.Media.TextFormatting 名字空间展现的例子成员。

类	
InlineObjectInfo	提供内部文本对象的测量细节。格式化客户将此对象作为一参数传送到 GetInlineObjectInfo 方法。
TextFormatter	TextFormatter 是“Avalon”文本引擎并提供格式化文本和对文本分行的服务。TextFormatter 可以处理不同的文本字符格式和段落样式，并包括对国际化文本布局的支持。
TextHighlightBounds	文本范围的边界。
TextInfo	表示有关客户的文本源字符存储中的文本块的信息。
TextLine	对一行文本提供服务。从这个类继承来实现处理和格式化一行文本的服务。这是一个抽象类。
TextMarkerGeneratedContent	生成行列表标记输出。
TextMarkerInfo	定义段落的列表标记的样式和类型。格式化客户将此类作为参数使用以向 GetTextMarkerInfo 方法提供标记细节。
TextParagraphProperties	表示可以从一个段落改变到下一个段落的属性，如流方向、对齐或缩进。
TextRun	定义共享单个属性集合的字符序列。格式化客户在 TextFormatter 将此类作为 GetTextRun 方法的参数传递时提供 TextRun 的细节到这个类。
TextRunBounds	文本范围的边界。
TextRunCache	向 TextFormatter 对象提供缓冲服务以提高性能。

TextRunClientData	表示和 TextRun 关联的客户信息数据。
TextRunProperties	提供可以从一个 TextRun 改变到另一个的属性，如字形或前景笔划。这是一个抽象类。
TextRunTypographyProperties	提供 TextRun 的字体属性。这个客户属性集合产生由 OpenType 布局引擎处理的特性集合。
TextSource	向 TextFormatter 提供字符数据及格式化属性。所有对 TextSource 中的文本的访问都通过 GetTextRun 方法实现，此方法设计成允许客户以它选择的任何方式可视化文本。这是一个抽象类。
TextTrimmingInfo	提供文本剪裁特性的描述。当 TextFormatter 将此类作为参数传递给 GetTextTrimmingInfo 方法时，格式化客户填入剪裁细节到这个类。

枚举	
TextParagraphFlags	描述段落特性的标志。
TextRunCacheFlags	文本范围缓存器中的内容的类型。
TextRunType	指示 TextRun 的类型。

结构	
TextSourceCharacterIndex	TextSourceCharacterIndex 表示文本中的插入记号或字符位置。

## 媒体集成层

媒体集成层(MIL)为开发者或程序员提供 API 使得可以在他们的应用中以直接的方式实现可能的复杂合成效果，而以不对正常的应用性能产生负面影响方式充分利用图形处理单元。MIL 的一个方面提供组合不同媒体类型(如，

2D、3D、视频、音频、文本、图像等等)并流畅无缝地一起用动画呈现它们的能力。

MIL 提供图形架构，用于多阶段合成及允许在编程和脚本接口进行功能类比的编程模型。API 和脚本允许创建保留的结构或在呈现时合成的场景描述，还包括具有更立即方式本质的区域。

通过接口，MIL 提供对存储可视信息的数据结构的访问，从而应用软件可以利用由计算机硬件提供的图像能力。接口基于元素对象模型和矢量图形标记语言，用于以允许程序代码开发者稳定地连接场景图形数据结构来产生图形的方式来使用该元素对象模型。也可以使用此数据结构进行直接呈现或“编译” 可视信息，从而可以给低层的图形系统提供快速的合成和动画。

矢量图形元素对象模型总的来说对应于形状元素和其他元素，包括和场景图形的场景图形对象模型相关的图像及视频元素。标记可以解析为数据，包括元素树中的元素，元素又可以转换为场景图形数据结构中的对象。其他标记可以直接转换为数据及创建场景图形对象的调用。标记语言提供独特的方式来描述元素，包括简单的字符串格式或复杂的属性语法，属性可被重命名，使能在标记中的其他地方进行重用。

MIL 的一个方面是跨 API 集合集成动画和定时，将动画提供作为内在的基础级别概念。为了帮助流畅的动画，MIL 提供(如操作系统的)多层次图处理系统和方法。一个那样的图形处理系统包含两个组件，这包括按命令运作或慢速运作的高层组件，及快速处理(如，以图形硬件帧刷新速率)的低层组件。总的来说，为了向低层组件传递简化的数据结构，高层、频率较低的组件执行计算量大的更新动画参数和穿过场景数据结构的方面。为了处理数据结构而生成恒定的图形子系统输出数据，低层组件以较高的频率，如图形子系统的帧刷新率工作。低层处理包括在需要获取实时值以呈现动画的每一帧的场景时，对任何参数时间间隔进行内值计算。

顶层 MIL 对象包括可视组件树，它是包含将画出的主要内容的对象。控件将直接从此树的可视组件导出。可视组件独立于设备和父组件上下文。呈现目标是画可视组件的设备。此对象，(如，屏幕)可以有它自身的缺陷或不足的机制。各种呈现目标包括屏幕上的窗口、打印机、元文件、表面及“子窗口”，子窗口为独立于剩下的场景画出的部分场景。其他和画图组件相关的对象包括可视组件呈现器，它包括配置为将可视组件树画到呈现目标上的

对象，及知道何时将可视组件树画到呈现目标上的显示调度器对象。时间管理器是一组定时节点的上下文对象，而且是调度器调用运作的对象。

提供可视 API，这本质上是通过媒体集成层画图组件的起始点，并包括多种类型的对象，包括 VisualManager 对象，它将 VisualTree 连接到媒体。不同类型的 VisualManager(如， Screen、Printer 和 Surface)负责呈现 VisualTree 到它们特定的媒体。可视组件是其中程序员画图的地方；它是可视组件树中的节点并为程序提供画图的地方。

DrawingContext API 提供基于上下文的编程模型，编程模型用于确定如何构建放置 Visual 的可视内容或将其呈现到 ImageData。提供 DrawingContext 类，及获取 DrawingContext 并枚举 RetainedVisual/DrawingVisual 中的可视内容所需的类和切入点。

为了支持可变性，提供了从 Changeable 基类导出的一组类型。任何需要可变性的类型都可以从 Changeable 类导出。例如，在图形编程中，对象模型包括 Brush、Pen、Geometry、FloatAnimation、GradientStop、Segment 等等。IsChangeable 属性根据它的定义一个状态的当前值规定是否可以修改可变的对象。

画笔是表示填充平面的方法的对象。除了能够以绝对的方式填充平面以外，媒体集成层的画笔也可以确定它们如何相对于它们所填充的对象的大小填充平面。画笔类型的例子包括 SolidColorBrush、VisualBrush(它可以引用矢量图形资源/Visual)、DrawingBrush、LinearGradient、RadialGradient、ImageBrush 和 NineGridBrush。某些的画笔对象可以在使用它们时知道它们和坐标系统的相对关系，也可以知道它们和它们所用的几何图形边框的相对关系。这个大小基于画笔所填充的对象。Brush 基类有转换、通用不透明度和混合模式。画笔(及矢量图形和 MIL API 中的其他对象资源)对象是 Changeable 且在它们被创建之后可以对它们进行写入，并遵从通用的 Changeable 模式以确定它们在以正常方式使用以后的行为。

Geometry 类的对象可以用于剪裁、点击测试及用 Pen 和 Brush 呈现基于矢量的 2D 数据。导出的 Geometry 类提供更加具体的构建和枚举语义。提供一些特定于形状的 Geometry 类型，以及允许明确定义有更加复杂的形状的 Geometry 的广义 PathGeometry。Geometry 是抽象基类。GeometryCollection 是在它们定义的区域上使用特殊的 CombineMode 操作组合的多个 Geometry 对

象的集合。此对象允许 Geometry 对象可视组件组合的创建比在 PathGeometry 内严格的使用 PathFigure 对象更加简单。

ImageSource 是抽象类，包括用于图像处理的基本构建单元。ImageSource 概念上表示在特定大小和分辨率下单个恒定的象素集合。例如，ImageSource 可以是 Decoder 可以提供的图形文件中的单个帧，或可以是对特定的 ImageSource 自身进行操作的转换结果。ImageSource 是可变的，不仅因为它自身的属性可以改变，也因为它的子类的属性也可能改变。

提供 Transform 类的对象用于缩放、旋转、平移和扭曲矢量和栅格图形。导出的 Transform 类提供友好的用法和枚举语义。

Effects 提供以呈现为中心的方式更改场景的可视内容的方法。例如，VisualEffects(基于栅格的位图效果)操作基于图像的、完全合成的部分场景的表示。Effects 划分为各种类型，包括 VisualEffects、BlendModes 和 VectorEffects。通过对子图形或元素应用 VisualEffects，它可以用在保留模式场景中，或它可以用在单独的图像管线中。BlendModes 是基于图像的效果的具体形式，且能够以和 VisualEffects 大致相同的方式将其应用于保留模式的场景。当源是合成的，如相乘或相加时，Blend 模式执行源和目标颜色的合并。

点击测试被用于挑选场景中的可视组件，并通过从控件树的顶部开始，并通过由一个点或几何形状返回一个控件或一组控件来进行操作。控件可以用支持服务确定它是否被点击，支持服务包括呈现的几何图形、边框、范围外的几何图形(点击区域)、图像不透明度或遮蔽，及其自身的逻辑。在点击到时，控件可以返回特定的有关点击的数据。点击测试机制可以用有效的方式过滤点击测试结果。点击测试途径是可视组件树中从右到左的很深的途径，通过以 z 顺序，从上到下的方式进行回调来报告点击。当向下降时，点击测试器借助元素层次关系查看过滤，例如，有形状的画布，或有内部画布的泊放面板。当发生点击时，点击测试器可以进一步继续处理点击(如果有的话)，或停止。

提供动画系统，它包括定时控件引擎和一组动画对象。定时引擎是可由具有随时间而变化行为的任何对象，如动画和音频或视频媒体对象使用的服务。动画对象实现将时间段跨度映射到其他数据类型然后将其用作其他高层对象的输入的一组功能。通过关联动画集合与呈现操作来实现图形动画。在

呈现操作中使用的每个动画可以按称为“时间线”的独立时钟来运行。一旦画出了动画原型且指定了动画参数，低层呈现系统以固定的时间间隔处理场景的重画。每次呈现帧时，就基于经过的时间（在多数情况下由系统时钟来测量）对包括在场景中的动画的当前值进行计算，然后重画该动画原型。

也通过 MIL 提供各种原语类型、颜色特性和媒体支持。MediaData 可以用来播放任何音频/视频内容。

### 设计名字空间

设计名字空间 320 提供支持表单和文本编辑、格式化数据及跨进程数据共享的类。这些类提供用于编辑文档、应用和其他内容的可扩展框架。设计名字空间 320 包含两个层次上的功能：面向需要预打包的对不同类型的信息的即刻可用的编辑器的应用开发者的高层功能；及面向引入它自身类型的数据的更加高级的应用的低层功能。作为示例的预打包的即刻可用的编辑器提供纯文本编辑、丰富格式文本编辑、表单（元素布局）编辑和密码输入。

设计名字空间 320 提供灵活且可扩展的方法来组织各种功能。设计名字空间 320 提供服务、行为和抽象的组合来允许开发者构建专门且易于客户化的信息编辑解决方案，而不是提供普通的固定特性的编辑器。设计名字空间 320 包括几个可定制的编辑特性，包括：范围服务、可堆放行为、编辑器—设计器模式、用于元素编辑的抽象设计器、抽象文本对象模型、修饰层、设计表面、通用数据传输协议，及可扩展的撤销机制。

有范围的服务允许将特定服务和应用数据的特定部分和子部分关联。此机制也允许在不同的范围内打开或关闭服务。提供服务支持的主要类是 ServiceManager；使用 IService 和 IScopeService 接口来实现单独服务。

“可堆放行为”的概念允许在适当的时间段激活不同的行为。在一个实施例中，嵌套基于时间的行为激活，使得在更一般的进程暂时挂起并随后恢复的同时可以开始和结束子进程。可堆放行为的方法通过允许集成预先对彼此不了解的进程解决了这个问题，而不采用硬编码解决方案。提供对此机制的支持的主要的类是 EditRouter；单独行为是 Editor 和 EditBehavior 类的子类。可编辑数据的内置类型由 EditBehavior 的对应子类支持，如 TextEditor、ElementEditor 和 MoveBehavior。

“编辑器—设计器模式”的概念允许分离通用的编辑功能和更加具体的

可编辑信息的子类型。编辑器不期望操作特定的数据结构。相反，它假设使用通过某些抽象接口展现数据内容，隐藏实现细节，但是对于编辑交互的对应动作已经足够了。此模式对元素（在表单编辑中）和文本（在丰富格式文本编辑中）特别有用。

“元素编辑的抽象设计器”是用于元素编辑（如，表单编辑）的领域的“编辑器—设计器模式”的应用。各种元素可以有不同的内部结构、外观和行为，但是如果它们展现某些用于移动、旋转或重设大小标准的“设计器”，则它们变得由元素编辑器可编辑。

抽象文本模型是用于丰富格式文本编辑的“编辑器—设计器模式”的另一个应用。此抽象允许不同的后备存储来参与丰富格式文本编辑，如 RTF、XHTML、纯文本、XAML 标记、按语法进行强调的源代码等等。即使这些文本类型的语义本质和内部结构有所不同，通过经抽象文本模型展现自身，它们支持通用编辑机制的应用。另外，此方法支持文本和数据集成，从而来自不同后备存储的文本可以包含彼此的片段，同时提供无缝的编辑操作。

修饰层是在编辑中提供丰富可视反馈的统一工具。修饰层对多种类型反馈提供统一的支持。强大的可视组件可以动态地与元素及数据的其他部分连接和分离来指出可能的作用于它们的用户活动。设计表面是预打包编辑资源（如修饰层和编辑路由器）的 XAML 控件。这简化了中间层次的编辑开发。

如剪切/复制/粘贴和拖/放这样的编辑功能在集成的信息环境中是很重要的，但是可能由于不同数据类型的复杂性和不兼容而难以支持。通用数据传输协议通过提供用于数据抽取、数据插入、数据拖动和数据转换的抽象组合来处理此问题。这个通用的数据传输协议对应用数据集成提供强大和灵活的机制。

可扩展的撤消机制通过收集各种类型的数据来管理撤消栈。撤消管理器包括处理具有不同数据类型的撤消操作的协议。

下面的表列出由 System.Windows.Design 名字空间展现的例子成员。

类	
ActivationEventFilter	事件过滤器。
ActivationRectangleAdornerInfo	这是示出 DesignActive 元素边界的反馈。

AdornerInfo	提供有关修饰器集合中的特定修饰器的信息。
AdornerLayer	提供用于显示需要超越 Z 顺序(元素总是需要在顶上)的元素上的修饰的表面。
CanvasDesigner	缺省的 Canvas 设计器。
ComponentChangeService	提供用于影响组件的设计时动作的撤消和重做功能。
Designer	提供编辑活动内容的方法。
DesignerFilterService	定义可选的服务，使应用和控制可以用它来实现覆盖由 DesignerLookupService 提供的元素和设计器之间的缺省映射。
DesignerLookupService	支持映射元素到设计器的编辑行为。
DesignSurface	推荐的对可编辑内容根元素。它聚合了修饰器、服务和事件。
EditBehavior	定义所有编辑行为的实现。
Editor	处理用户对特殊选择类型，如文本、元素或表的输入。通常对每种选择类型都有特定的编辑器。
EditorTypeAttribute	关联选择类型和选择模式。不能从这个类继承。
EditRouter	管理当前活动的 EditBehavior 的集合。
EditRouterChangedEventArgs	Changed 事件的参数。
ElementEditor	元素编辑器。
EventFilter	支持事件过滤和分配。
FrameworkElementDesigner	对所有从 FrameworkElement 导出的控件的缺省 Designer。
MoveBehavior	移动行为。
ObjectSelection	位置选择。

ParentUndoUnit	ParentUndoUnit
RichTextDesigner	TextPanel 和支持 ITextDocumentResult 的其他控件的 ITextDesigner 实现。
RoutedEventAdapter	RoutedEventAdapter 将 特 定 的 eventHandler 向 下 转 换 为 RoutedEventHandler 从 而 一 个 处 理 程 序 可 以 处 理 不 同 类 型 的 事 件 。 由 EditRouter 使用。
SelectionRouter	SelectionRouter 控 制 与 多 种 选 择 模 式 的 混 合 选 择 。 它 从 EditRouter 导 出 以 路 由 事 件 到 模 块 编 辑 器 ( 它 从 Editor 导 出 ) 。
SelectionService	提 供 对 UI 项 的 选 择 的 编 程 访 问 。 可 以 在 选 择 的 项 上 获 得 反 馈 ， 并 更 改 选 择 的 项 及 其 属 性 。
ServiceConverter	ServiceConverter 一 用 于 在 字 符 串 和 IService 的 实 例 之 间 进 行 转 换 的 转 换 器 类 。
SimpleTextDesigner	用 于 Text 和 支 持 ITextParagraphResult 的 其 他 控 件 的 ITextDesigner 实 现 。
TextDecorator	TextDecorator 定 义 管 理 来 自 竞 争 服 务 的 文 本 修 饰 的 对 象 。 使 用 TextDecorator 而 非 私 自 写 入 属性 值 来 确 保 竞 争 的 修 饰 将 永 不 重 叠 ， 甚 至 在 有 相 同 优 先 权 的 情 况 下 。
TextEditor	提 供 文 本 选 择 服 务 。
TextSelection	TextSelection 类 封 装 对 TextEditor 类 的 选 择 状 态 。 它 没 有 公 共 的 构 造 器 ， 而 是 通 过 TextEditor 上 的 公 共 属 性 展 现 。
UndoService	IUndoService 的 框 架 实 现 。
UndoStackChangedEventArgs	向 UndoStackChanged 和

	RedoStackChanged 事件提供数据。
--	--------------------------

接口	
IAddRemoveChildDesigner	IAddRemoveChildDesigner 接口用于增加/移除子元素。
IAdornerDesigner	提供对支持检索在 FrameworkElement 上画出修饰器所需的对象的方法的访问。
IDesignActivationContainer	DesignActivation 中可以变为 designActive 的内部元素。具有名字组件，因为假设如果元素是 DesignActive，那么它很可能正在编辑某些子内容。
IDesignActivationContainerParent	IDesignActivationContainerParent 可能是设计环境中的宿主 IDesignActivationContainer。
IDesignSurface	由希望参与编辑的任何元素实现的接口。
IMoveDesigner	移动设计器接口。
IParentUndoUnit	IParentUndoUnit 接口。
IScopedService	IScopedService。
ISelection	选择模式的基础接口。
ISelectionDesigner	和 SelectionBehavior(SelectionRouter) 关联的设计器接口。实现此接口的类可以在元素类型的 Designer 属性中指定。
IService	这是用于编辑服务的 TypeConverter 的占位者。
ITextDesigner	向 TextEditor 服务提供特定于布局的信息和动作的接口组件实现。

IUndoService	IUndoService 接口。
IUndoUnit	IUndoUnit 接口。

枚举	
AdornerInfoZOrder	Z 顺序标志。
AdornerRenderTriggers	指出使修饰器重新呈现的条件的标志。
EditRouterChange	可能的路由改变动作的枚举。
TextDecorationPriority	TextDecorationPriority 对各种文本标记的相对重要性进行排序。它用于 TextDecorator 类中的 Z 顺序修饰。
UndoState	撤消管理器的状态的枚举。

代理	
UndoStackChangedEventHandler	表示处理 UndoStackChanged 或 RedoStackChanged 事件的方法。

### 输入名字空间

输入名字空间 322 包括协调由系统接收的输入的输入管理器。输入名字空间 322 也包括帮助管理不同的输入设备，如键盘或鼠标并对其提供控制的类。输入系统允许应用从如鼠标、键盘和触针(笔)这样的输入设备取得信息。多数输入功能以 System.Windows 名字空间中的 UIElement、FrameworkElement、ContentElement 和 ContentFrameworkElement 类的属性、方法和事件的形式存在。UIElement 和 ContentElement 有类似的输入功能，这由 IInputElement 接口获取。类似地，FrameworkElement 和 ContentFrameworkElement 分别从 UIElement 和 ContentElement 导出，它们共享很多输入 API 且两者都实现 IFrameworkInputElement。

输入系统提供对键盘、鼠标和触针的完整支持。键盘功能包括键按下/键弹起事件、焦点管理和通知、访问当前的键状态，和转换击键为输入字符串。鼠标功能包括鼠标位置、点击、移动、在元素边界件的进入/离开事件、鼠标捕捉、悬停和鼠标滚轮。触针功能包括位置和移动(笔接触表面和“在空中”均可)、轻按/点击、捕捉和姿势识别。

输入名字空间包含上述功能所需的辅助器类，如枚举、事件参数、代理签名等等。另外，输入名字空间包括 Keyboard、Mouse 和 Stylus 类，它们提供有关那些设备的当前状态的信息。输入系统提供方法，指定在特殊元素上以及对整个应用的鼠标光标。

输入系统和文本服务框架 (TSF) 集成，TSF 允许其他输入软件理解输入新文本到其中的文本环境。这由输入方法编辑器 (IME) 使用，IME 允许东亚语言的用户转换多个击键为单个字符—IME 软件和 TSF 通讯以基于上下文识别击键的最佳翻译并使用 TSF 来插入文本到文档中。类似地，语音识别使用 TSF 来挑选最佳识别并将其插入文档。

输入系统以 TextInput 事件的形式提供与 TSF 的自动集成。输入名字空间也提供 API 用于描述文本环境，及控制从原始键盘输入到文本输入的转换。InputManager 类提供过滤器和监控器，它们允许第三方观察输入流并在触发初始输入事件之前创建新的输入事件。InputManager 也使用 IInputProvider 和 InputReport 提供用于被插入输入系统的新设备的 API。

下面的表列出由 System.Windows.Input 名字空间展现的例子成员。

类	
AccessKeyManager	AccessKeyManager 展现访问键功能，也称为助记符。典型例子是 Ok 键，Alt-O 作为它的助记符—按下 Alt-O 和点击该按钮效果相同。
Cursor	表示所使用的鼠标光标。
CursorTypeConverter	在鼠标光标和字符串之间进行转换。
FocusChangedEventArgs	FocusChangedEventArgs 类包含有关焦点改变的信息。
InputDevice	提供所有输入设备的基础类。
InputEventArgs	InputEventArgs 类表示和所有的输入事件有关的一种 RoutedEventArgs
InputLanguageChangedEventArgs	InputLanguageChangedEventArgs 类表示和指示(人的)输入语言的改变而触发的事件有关的一种 RoutedEventArgs。

InputLanguageChangingEventArgs	InputLanguageChangingEventArgs 类表示和指示(人的)输入语言的改变而触发的事件有关的一种 RoutedEventArgs。
InputLanguageEventArgs	InputLanguageEventArgs 类表示和指示(人的)输入语言的改变而触发的事件有关的一种 RoutedEventArgs。
InputLanguageManager	InputLanguageManager 类负责管理 Avalon 中的输入语言。
InputManager	InputManager 类负责协调“Avalon”中的所有输入系统。
InputMethod	InputMethod 类展现有关 TSF 的 API, 它发送或访问 TIP 的属性。
InputMethodStateChangedEventArgs	当输入方法编辑器(IME)改变其状态时使用此 InputMethodStateChangedEventArgs 类。
InputProviderSite	输入提供者用来报告对输入管理器的输入的对象。
InputReport	InputReport 是向 InputManager 报告的所有输入的抽象基础类。
InputReportEventArgs	提供在处理 InputReport 时触发的事件的数据。
Keyboard	Keyboard 类展现涉及键盘的 API。
KeyboardDevice	KeyboardDevice 类表示对上下文成员的键盘设备
KeyboardEventArgs	KeyboardEventArgs 类对所有导出的事件的参数提供对逻辑指针设备的访问。
KeyEventArgs	KeyEventArgs 类包含有关键状态的信息。
KeyInterop	提供在 Win32 虚拟键和“Avalon”键枚

	举之间进行转换的静态方法。
Mouse	Mouse 类展现涉及鼠标的 API。
MouseButtonEventArgs	MouseButtonEventArgs 描述鼠标按钮的状态。
MouseDevice	MouseDevice 类表示对上下文成员的鼠标设备。
MouseDoubleClickEventArgs	提供双击鼠标时触发的事件的数据。
MouseEventArgs	MouseEventArgs 类对所有导出的事件的参数提供对逻辑鼠标设备的访问。
MouseWheelEventArgs	MouseWheelEventArgs 描述鼠标滚轮的状态。
NotifyInputEventArgs	提供有关由输入管理器处理的输入事件的信息。
PreProcessInputEventArgs	允许处理程序取消对输入事件的处理。
ProcessInputEventArgs	提供对输入管理器的分级区域的访问。
RawKeyboardInputReport	RawKeyboardInputReport 类封装从键盘提供的原始输入。
RawMouseInputReport	RawMouseInputReport 类封装从鼠标提供的原始输入。
StagingAreaInputItem	这个类封装在由输入管理器处理时的输入事件。
TextInputEventArgs	TextInputEventArgs 类包含输入的文本表示。
TextManager	TextManager 类提供输入到文本事件提升。
TextServicesContext	这个类管理 UIDispatcher 和文本服务框架之间的互操作，使能作东亚语言 IME 输入的本地 COM API。
TextStoreInfo	这是内部的链接需求保护类。

接口	
IInputLanguageSource	控制输入语言源的接口。
IInputProvider	由所有输入提供者实现的接口。
IKbdInputProvider	用于控制键盘输入提供器的接口。
IMouseInputProvider	用于控制鼠标输入提供器的接口。

枚举	
CursorType	所支持的光标类型的枚举。
ImeConversionMode	ImeConversionMode
ImeSentenceMode	ImeSentenceMode
InputMethodState	Ime 的状态。
InputMode	当提供输入时的输入处理的模式。
InputType	所报告的输入的类型。
Key	键盘上所有可能的键值的枚举。
KeyState	KeyState 枚举描述键盘上的键可能所处的状态。
ModifierKeys	ModifierKeys 枚举描述用于修改其他输入操作的一组通用键。
MouseButton	MouseButton 枚举描述鼠标设备上的可用按钮。
MouseButtonState	MouseButtonState 枚举描述鼠标输入设备上可用按钮的可能状态。
RawKeyboardActions	从键盘报告的原始动作。
RawMouseActions	从鼠标报告的原始动作。
SpeechMode	语音模式。

结构	
TextServicesMSG	Win32 MSG 结构的管理的版本。

代理	
----	--

FocusChangedEventHandler	用于接收 FocusChangedEventArgs 的处理程序的代理。
InputEventHandler	用于接收 InputEventArgs 的处理程序的代理。
InputLanguageEventHandler	对 InputLanguageChanged 和 InputLanguageChanging 事件的代理。
InputMethodStateChangedEventHandler	用于接收输入方法状态改变事件的处理程序的代理。
InputReportEventHandler	用于接收 PointerMoveEventArgs 的处理程序的代理。
KeyboardEventHandler	用于接收 KeyboardEventArgs 的处理程序的代理。
KeyEventHandler	用于接收 KeyEventArgs 的处理程序的代理。
MouseButtonEventHandler	用于接收 MouseButtonEventArgs 的处理程序的代理。
MouseDoubleClickEventHandler	用于接收 MouseDoubleClickEventArgs 的处理程序的代理。
MouseEventHandler	用于接收 MouseEventArgs 的处理程序的代理。
MouseWheelEventHandler	用于接收 MouseWheelEventArgs 的处理程序的代理。
NotifyInputEventHandler	用于处理使用 NotifyInputEventArgs 的事件的代理类型。
PreProcessInputEventHandler	用 于 处 理 使 用 PreProcessInputEventArgs 的事件的代理类型。
ProcessInputEventHandler	用于处理使用 ProcessInputEventArgs 的事件的代理类型。
TextInputEventHandler	用于接收 TextInputEventArgs 的处理

	程序的代理。
--	--------

## 导航名字空间

导航名字空间 324 提供允许构建有浏览方案的应用，如浏览器应用的一组类和服务。这些类和服务允许开发带有客户化的导航操作的应用。例如，当从网上商家购买产品或服务时，点击“返回”按钮使得应用显示不同的页面，询问用户是否希望取消或改变他们的订单。在另一个例子中，激活“刷新”按钮使得应用软件检索新的数据而不是首先重新加载该应用软件接着再检索新数据。导航名字空间 324 也包括用于提供产生向用户提供的问题层次的机制的页函数。

下面的表列出由 System. Windows. Navigation 名字空间展现的例子成员。

类	
BoolWrapper	Boolean 值的包装器。
BoolWrapperPageFunction	返回 Boolean 值到前一页的有类型的页函数。
BoolWrapperEventArgs	
IntWrapper	Int32 值的包装器。
IntWrapperPageFunction	返回 Int32 值到前一页的有类型的页函数。
IntWrapperEventArgs	
Journal	包含应用软件的导航历史。
JournalEntry	表示日志条目。
LoaderService	用来在给定的应用领域中设置当前的加载器。
NavigateEventArgs	已废弃。
NavigatingCancelEventArgs	对 NavigationApplication. Navigating 和 NavigationWindow. Navigating 事件的事件参数。
NavigatingNewWindowCancelEven	可取消事件 NavigatingNewWindow 的事

tArgs	件参数。 NavigatingNewWindowCancelEventArgs 指定目标 NavigationContainer，在其中用传入的 URI 或元素进行导航。缺省的 Cancel 属性设置为假。设置 Cancel 为真将防止打开新窗口，且不进行导航。
NavigationApplication	表示对系统的“Longhorn”导航应用。
NavigationContainer	可以包含“Avalon”标记树的可浏览区域。通常，不直接使用这个类但是可以将其用作对客户实现的父类。
NavigationErrorCancelEventArgs	包含对 NavigationApplication. NavigationError 和 NavigationWindow. NavigationError 事件的参数。
NavigationEventArgs	不可取消导航事件，包括 LoadCompleted 、 LoadStarted 、 Navigated 和 NavigationStopped 的事件参数。
NavigationProgressEventArgs	包含对 NavigationApplication. NavigationProcess 和 NavigationWindow. NavigationProcess 事件的参数。
NavigationService	包含由导航事件使用的代理和包含 INavigator 接口的动态属性。
NavigationWindow	表示一导航窗口。
ObjectPageFunction	返回 Object 值到前一页的有类型的页函数。
ObjectReturnEventArgs	

PageFunction	不直接支持这个类。而是使用下面有类型的类之一： BoolWrapperPageFunction 、 IntWrapperPageFunction 、 ObjectPageFunction 或 StringPageFunction。
ReturnArgs	对返回事件的事件参数对象。不直接支持这个类。而是对适合的有类型的类使用返回参数，这些类是： BoolWrapperEventArgs 、 IntWrapperEventArgs 、 ObjectEventArgs 或 StringEventArgs。
StringPageFunction	返回 String 值到前一页的有类型的页函数。
StringEventArgs	
WindowNavigationContainer	表示导航窗口内的可导航区域。

接口	
IJournalData	IJournalData 接口一应由需要在日志中持续状态的控件实现，并在重新访问该页时恢复它。
ILoader	用于分解 URI 为流的接口。它可以用于从文件、http、组件及管理的和未管理的资源加载内容。
INavigator	由浏览器实现以提供对支持导航的属性、方法和事件的访问。
INavigatorService	INavigatorService 接口。此接口在支持 INavigator 任何的实现者注册它们自己以参与 Hyperlink 指向的 NavigationWindow 上可用。

枚举	
NavigationMode	用于指定导航模式。

代理	
BoolWrapperReturnEventHandler	
IntWrapperReturnEventHandler	
LoadCompletedEventHandler	表 示 处 理 NavigationApplication. LoadCompleted 和 NavigationWindow. LoadCompleted 事件的方法
LoadStartedEventHandler	表示处理 LoadStart 事件的方法。
NavigatedEventHandler	表 示 处 理 NavigationApplication. Navigated 事件和 NavigationWindow. Navigated 事件的方法。
NavigateEventHandler	已废弃。
NavigatingCancelEventHandler	表 示 处 理 NavigationApplication. Navigating 和 NavigationWindow. Navigating 事件的方法。
NavigatingNewWindowCancelEventHan dler	表示处理 NavigatingNewWindow 事件的方法。
NavigationErrorCancelEventHandler	对 NavigationApplication. NavigationEr ror 事 件 和 NavigationWindow. NavigationError 事 件 使用此代理。
NavigationProgressEventHandler	表 示 处 理 NavigationApplication. NavigationPr

	ogress 和 NavigationWindow.NavigationProgress 事件的方法。
NavigationStoppedEventHandler	表示处理 NavigationApplication 对 NavigationWindow.NarigationProgress 和 NavigationApplicationNarigationProgress 事件 NavigationStopped 和 NavigationWindow.NavigationStopped 事件的方法。
ObjectReturnEventHandler	
ReturnEventHandler	表示处理 Return 事件的方法。不直接支持这个类。而是使用适当的有类型的类的返回事件处理程序，这些类是： BoolWrapperReturnEventHandler 、 IntWrapperReturnEventHandler 、 ObjectReturnEventHandler 或 StringReturnEventHandler。
StringReturnEventHandler	

## 自动化名字空间

自动化名字空间 326 包含用于支持可访问性及用户接口自动化的成员。可访问性系统包括客户端和提供者端。工具的系统包括客户端自动化工具，客户端自动化工具包含用于搜索用户接口信息的客户自动化类。客户自动化类包括事件注册工具和逻辑元素发现工具。这组工具进一步包括用于向客户提供用户接口信息的提供者端自动化工具。提供者端自动化工具包括用于向客户提供事件信息的工具的自动化提供者类。

客户自动化类提供用于一个或多个客户的 UI 自动化方法。客户自动化类包含不特定于任何 UI 元素的方法。客户自动化类可以提供从点、窗口句柄或桌面根元素获取逻辑或原始元素的方法。客户自动化类还可以提供基于输入准则的查找逻辑元素的方法。客户自动化类最好还包括注册和注销事件通知

的方法。自动化类最好还提供用于加载代理 DLL、检索本地化的属性名称和控件模式并执行元素比较的辅助函数。客户自动化类也包括由客户用来侦听事件的方法。下面的表列出由 System.Windows.Automation 名字空间展现的例子成员。

类	
ApplicationWindowPattern	展现通常和顶层应用窗口关联的行为和信息。客户可以使用这个类来贴图或级联应用的多文档接口 (MDI) 子元素，在任务栏上找出其按钮，并定位用户接口众所周知的部分，如工具栏和菜单。
AssociatedInformationPattern	展现表示其他对象的 UI 元素的语义和元数据。
Automation	包含对客户的用户接口 (UI) 自动化方法（辅助技术或自动化测试脚本）。这些方法并不特定于特殊的元素。
AutomationEvent	请勿使用。这个类将在未来版本中去除。
AutomationEventArgs	模式或定制事件参数类。
AutomationFocusChangedEventArgs	焦点事件参数类。
AutomationIdentifier	对象的基于身份标识符的基类。这个类实际上是抽象的；仅实例化导出的类。
AutomationIdentifierProxy	用于反序列化 AutomationIdentifier 的内部类。请勿直接使用。
AutomationPattern	请勿使用。这个类将在未来版本中去除。
AutomationPermission	提供一组访问 UI 元素的许可。不能从这个类继承。
AutomationPermissionAttribute	为 AutomationPermissionAttribute 提供方法和属性。不能从这个类继承。

AutomationProperty	请勿使用。这个类将在未来版本中去除。
AutomationPropertyChangedEventArgs	PropertyChanged 事件参数类。
AutomationTextAttribute	自动化文本属性的标识符。
AutomationTextPointer	表示字符在文本中的位置。 AutomationTextPointer 提供访问文本和进行文本导航的方法和属性。
AutomationTextRange	用于获取、设置、增加和移除选择。
AutomationTextSelection	目的： AutomationTextSelection 对象处理所有的文本选择管理。插入指针有效的选择是活跃的选择。例子用法：当客户希望增加、移除、修改或设置选择时使用。客户也可以通过 AutomationTextSelection 找出当前选择了什么。
AutomationTextUnit	自动化文本单位的标识符。
BasePattern	内部类。
DockPattern	展现元素在运行时改变其泊放状态的能力。
ElementNotEnabledException	这个异常在客户代码尝试处理当前未启用的元素或控件时发出。
ElementPath	ElementPath 提供接下来返回到原先由应用提供者记录、修改、或完全创建的逻辑元素所需的准则。
ExpandCollapsePattern	展现控件来显示更多内容或折叠以隐藏内容的扩展能力。例子包括菜单按钮、开始按钮、Windows 资源管理器中的树形视图项及组合框。
GridItemPattern	展现网格的元素。允许客户快速确定网

	格项的坐标。
GridPattern	展现基本的网格功能、大小和导航到指定单元格。
HierarchyItemPattern	独立于它们在逻辑树中的关系，展现控件的用户接口元件之间的层次关系。
Input	提供发送鼠标和键盘输入的方法。
InvokePattern	由执行单个明确的动作的对象(控件)实现。这些控件中大多数不保持状态；引用它们起动应用中的动作。实现此接口的控件的例子包括按钮、工具栏按钮、菜单项、超级链接、复选框、单选按钮、树形视图控件中的加号及 Windows 资源管理器中的列表视图项。
LogicalElement	表示逻辑树中的 UI 元件而不管它的实现 (“Avalon”、Microsoft® Win32®)。
LogicalStructureChangedEventArgs	逻辑结构改变事件参数类。
MultipleViewPattern	展现元素在同一组信息、数据或子元素的多种表示之间切换的能力。例子包括 ListView(缩略图、贴图、图标、列表、细节)、Excel 图表(饼图、直线图、直方图、带有公式的单元格的值)及 Outlook 日历(年、月、周、日)。
NoClickablePointException	当参数的值超出如由 GetClickablePoint 定义的允许的范围值之外时发出此异常。例如，当界定的矩形为空，在该点上没有宽度或高度或 LogicalElement，这和所调用的不同。
ProxyAssemblyNotLoadedException	此异常在加载代理组合出现问题时发出。这可能发生在响应 Automation.RegisterProxyAssembly

	时，或在碰到第一个 hwnd 基础 LogicalElement 时加载缺省代理的情况发生。
RangeValuePattern	展现控件管理有限范围内的值的能力。它传递控件的有效最小和最大值及当前值。
RawElement	表示原始元素树中的元素。
ScrollPattern	表示表达一值的 UI 元素。
SelectionItemPattern	表示管理选择的组件内的单个项。UI 自动化客户使用这个类来得到有关支持 SelectionItemPattern 控件模式的 UI 元素的信息或对其进行处理。例子包括任何可以在支持 SelectionPattern 控件模式的控件中选择的项，如列表框或树形视图中的项。
SelectionPattern	表示管理选择的组件。由 UI 自动化客户 使用 以 得 到 有 关 支 持 SelectionItemPattern 控件模式的用户接口元素的信息或对其进行处理。
SortPattern	展现组件的当前排序顺序并允许客户接 程序地重排元素。
SplitPattern	表示可以通过创建邻接的窗口克隆它们自己的窗口。
TableItemPattern	表示有标题信息的网格项。
TablePattern	表示有标题信息的网格。
TextPattern	表示文本，如编辑控件。
TopLevelWindowEventArgs	TopLevelWindowEventArgs 事件参数类。
TreeLoadEventArgs	TreeLoadEventArgs 事件参数类。
ValuePattern	展现不跨范围的控件的值，如单选按

	钮、二选一按钮、复选框、编辑框、RGB 颜色值和可复选的菜单项。
VisualInformationPattern	展现向用户传达信息的图形或动画有关的信息。这样的图形或动画如 IE 浏览器中指示正在下载文档的飘动旗帜或旋转地球或 Windows 资源管理器中指示正在进行复制的飞行纸片。
WindowPattern	展现元素改变其在屏幕上的位置或大小的能力。例子包括顶层应用窗口 (Word 或 Windows 资源管理器)、Outlook 的主窗口窗格 (文件夹、电子邮件消息、任务) 及 MDI 子窗口。
ZoomPattern	展现控件中的当前缩放水平并允许客户按程序改变它。

枚举	
AutomationPermissionFlag	包含 AutomationPermission 对象的访问标志。
ChildArrangement	安排子元素的不同方法。
DockPosition	可泊放窗口所附着的容件边缘。
ExpandCollapseState	由 ExpandCollapse 模式使用来指出扩展的/折叠的状态。
GetDataOptions	GetData 的选项。
HierarchyItemState	指出层次项状态的枚举，层次项状态包括：折叠、扩展或部分扩展。
ItemCheckState	指出复选框、单选按钮和类型控件状态的枚举。
LoadState	树状态标志。
LogicalMapping	指出是否在逻辑树中显示原始树中的节点的值。

ProxyFlags	用于指出请求属性的结果的枚举。
RowOrColumnMajor	在表中的数据最好按行表示还是按列表示。
ScopeFlags	在侦听事件时用于定义范围的标志。
ScrollAmount	由 ScrollPattern 使用来指示滚动方向(向前或向后)和滚动量(页或行)。
SendMouseInputFlags	SendMouseInput 的标志。这些标志表明是否发生了移动或是否按下或释放了按钮。
SplitStyles	分裂窗口的方向的标志。
StructureChangeType	指出逻辑树结构改变的标志。
TextPointerGravity	Gravity 确定当在 AutomationTextPointer 插入文本时 AutomationTextPointer 的位置。Gravity 总是从文本指针的视角出发, 即, 我在我的字符前面或我在我的字符后面。当用户在文本指针的位置插入字符时, Before 意味着指针将放在新字符的前面; After 意味着指针将放在新字符的后面。
WindowChangeType	指出顶层窗口的改变的标志。
WindowInteractionState	用于用户交互的窗口的当前状态。
WindowVisualState	用来描述窗口可视状态的状态。WindowVisualState 遵循 Office 和 HTML 对 WindowState 的定义。

结构	
MatchCondition	指定使用 FindLogicalElement 或 FindRawElement 查找元素的准则。
NameAndData	包含作为对象的名字、数据及作为字符

	串的数据。
ProxyDescription	包含有关代理的信息的结构。
SortInformation	用来排序的信息。

代理	
AutomationEventHandler	处理 AutomationEvent 的代理。
AutomationFocusChangedEventHandler	处理焦点改变事件的代理。
AutomationPropertyChangedEvent Handler	处理自动化属性改变事件的代理。
LogicalStructureChangedEventH andler	处理逻辑结构改变事件的代理。
ProxyFactoryCallback	由 HWND 处理程序实现；由 UI 自动化框架调用来请求对指定窗口的代理。如果支持的话应返回一代理，或在不支持的情况下返回空。
TopLevelWindowEventHandler	处理顶层窗口事件的代理。

下面的表列出由 System.Windows.Automation.InteropProvider 名字空间展现的例子成员。

类	
AutomationInteropProvider	包含由实现用户接口 (UI) 自动化的 Win32 应用或控件使用的方法。不能从这个类继承。
TextPointerPair	表示连续的字符块。

接口	
IApplicationWindowInteropProvid er	展现通常和顶层应用窗口关联的行为和信息。

IAssociatedInformationInteropProvider	展现表示其他对象的 UI 元素的语义和源数据。
IDockInteropProvider	展现元素在运行时改变其泊放状态的能力。
IExpandCollapseInteropProvider	展现扩展来显示更多内容或折叠以隐藏内容的控件能力。和 TreeView 项上的 HierarchyItem 模式相结合受到支持来提供类似于树的行为，但是也与打开和关闭的单个控件相关。实现此能力的 UI 例子包括：TreeView 项、折叠的 Office 的智能菜单、工具栏上的人字纹、组合框、菜单、Windows 资源管理器的任务栏的“扩展项”（通常显示文件夹视图的左边）。
IGridInteropProvider	展现基本的网格功能：改变大小和移动到指定单元格。
IGridItemInteropProvider	表示网格内的项。没有方法，只有属性。
IHierarchyItemInteropProvider	独立于它们在 the 逻辑树中的关系展现并允许客户穿过 UI 元素之间的层次关系。层次关系从定义上来说就是非循环的。 实现此功能的 UI 例子包括：TreeView 项、Viso 组织结构图、菜单及在“以分组显示”方式有效时的 Listview 控件。
IInvokeInteropProvider	由执行单个明确动作的对象（控件）实现。这些控件中的大多数不保持状态；引用它们起动应用中的一个动作。 实现此接口的用户接口元素的例子包括按钮、工具栏按钮、菜单项、超级链接、复选框、单选按钮、树形视图控件中的加号，及 Windows 资源管理器中的列表

	视图项。
IMultipleViewInteropProvider	展现元素在同一组信息、数据或子元素的多种表示之间切换的能力。此模式应在控制内容当前视图的组件上实现。
IRangeValueInteropProvider	展现反映控件管理有限范围内的值的能力的一组相关属性。它传递控件的有效最小值和最大值及其当前值。例子：数字旋 转 器 进 度 条 (NumericSpinnerProgressBar)、(在单个八位组上的)IP 控件、某些拾色器滚动条 (ColorPickersScrollBar)、某些滑动器的公共接口，它们表示表达当前值和值范围的 UI 元素。公共接口具有和 IValueProvider 相同的定义。两个模式的区别是 RangeValue 有附加属性，属性通常不出现在模式公共接口中。
IRawElementProviderFragment	由提供者实现以展现作为深度不止一层的结构的部分的元素。对没有子元素的简单单层结构，可以使用 IRawElementProviderSimple。片段的根 节 点 应 支 持 IRawElementProviderFragmentRoot 接口，它从此接口导出，并具有一些附加的方法。
IRawElementProviderFragmentRoot	UI 片段中的根元素应支持此接口。同一片段中的其他元素需要支持 IRawElementProviderFragment 接口。
IRawElementProviderFragmentRoot AdviseEvents	在 UI 片段的根元素上实现，以允许它在需要它来触发自动化事件时得到通知。

IRawElementProviderHwndOverride	由需要提供有关所包含的基于 HWND 的元素的信息或需要对其进行重定位的提供者实现。
IRawElementProviderOverrideType	由需要提供有关所包含的基于 HWND 的元素的信息或需要对其进行重定位的提供者实现。
IRawElementProviderSimple	UIAutomation 提供者接口，由希望展现单个元素属性的提供者实现。为了展现多于一个元素的属性和接口，参见导出的 IRawElementProviderFragment 接口。
IScrollInteropProvider	Scroll 模式展现控件通过滚动其内容改变其对用户可视的可视区域部分的能力。例子 Listbox、TreeView 及维持大于控件可视区域的内容区域的其他组件。注意滚动条自身应不支持 Scrollable 模式；它们支持 RangeValue 模式。服务器应归一化滚动(0 到 100)。这个公共接口表示滚动其内容的 UI 元素。
ISelectionInteropProvider	提供对由非“Longhorn”提供者在管理选择的组件上实现的属性的访问。
ISelectionItemInteropProvider	提供对定义和处理组件中选择的项的方法和属性的访问。仅在作为支持 ISelectionInteropProvider 且它自身是可选择的元素的子元素的逻辑元素上支持此接口。
ISortInteropProvider	展现组件的当前排序顺序并允许客户程序地对其元素重排序。某些组件在插入新项时保持排序顺序，或基于更新的内容移动其内容(例如，排序的 Win32

	列表框)。其他容件只能够执行一次性排序那在插入新项目时已过时(例如，Excel)。
ISplitInteropProvider	展现将元素内容区域分裂为多个格子或交互区域的能力。
ITableInteropProvider	标识有标题信息的网格。
ITableItemInteropProvider	用来展现有标题信息的网格项。
ITextInteropProvider	Text 模式展现控件处理文本的能力。例如： TextBox、RichEdit 控件及包含文本和文本相关属性的其他容件。此接口表示保持文本的 UI 元素。
ITextPointerInteropProvider	Text 模式展现控件处理文本的能力。例如： TextBox、RichEdit 控件及包含文本和文本相关属性的其他容件。此接口表示保持文本的 UI 元素。
IValueInteropProvider	表示表达一值的 UI 元素的公共接口。
IVisualInformationInteropProvider	用来表达有关向用户传达信息的图形或动画的信息。
IWindowInteropProvider	展现元素改变其在屏幕上的位置或大小及改变可视状态和关闭它的能力。
IZoomInteropProvider	展现控件中的当前缩放水平并允许客户按程序改变它。

枚举	
NavigateDirection	导航 UIAutomation 树的方向。
RawElementProviderOverrideType	指出此提供者是否作为覆盖者或是作为非客户区域提供者。

## 序列化名字空间

序列化名字空间 328 提供从 XML 文件或二进制表示的文件中加载对象(如

元素)的层次结构或将对象的层次结构保存到其中的解析器。此处理也设置和对对象相关的属性并关联诸事件处理程序。进一步来说，序列化名字空间 328 提供序列化和反序列化对象的功能。序列化是采用对象的活动的存储器中的表示，并产生适合于保存到存储设备(如硬盘)或通过网络发送的数据流。反序列化是反向的处理：即，取得数据流并从该数据流产生对象。反序列化亦称为“解析”，且实现解析的代码被称为解析器。

序列化和反序列化两者都支持多种流格式，如称为 XAML 的 XML 格式或称为 BAML 的二进制格式。当解析器和标记编译器一起使用时，它可以产生“CAML”，其中输出是产生序列化对象的代码。

序列化和反序列化支持全面的 XAML 特性，包括：标准语法、复合属性、紧凑的复合语法、隐含的子元素语法、显式的子元素语法、有明确集合标签的集合语法，及有隐含集合标签的集合语法。序列化和反序列化两者都是完全可扩展的。类可以通过下面几种机制定义它们希望如何序列化和反序列化，如：

- 缺省地，类名称作为标记标签名称处理，而属性和事件作为同一名称的标记属性处理。
- 类定义类型转换器(从 System.ComponentModel.TypeConverter 导出的辅助对象)以便序列化和反序列化标记属性值。
- 类可以通过实现 IAddChild 接口(解析)和使用 CLR 属性支持隐含的子类以描述如何 序列化隐含的子类。
- 类可以通过使用 DesignerSerializationVisibility 控制是否可以序列化 / 反序列化属性，并通过使用 DefaultValueAttribute 和 ShouldPersist 方法控制是否需要序列化特殊对象上的属性。
- 或者，组件可以使用字面内容序列化/反序列化其自身，在此情况组件的序列化形式不必符合 XAML 的规则(组件必须为格式良好的 XML)。

虽然上述例子使用了单词“标记”，同样的概念可以应用于二进制格式(BAML)。

第二个可扩展的机制是“设计器挂钩”，它允许插入另一段代码到序列化器和反序列化器中，并在解析和序列化过程中插入和截取属性。此机制允许创建新的标记特性，并允许现有语法的客户处理。标记编译器使用设计者挂钩来产生 CAML。

序列化和反序列化支持各种对象(如, 任何从 System.Object 导出的类)。从 System.Windows.DependencyObject 导出的类可以定义依存关系属性, 它们得到两个额外特性:

- 序列化可以自动地分辨是否需要序列化属性(对 DefaultValueAttribute/ShouldPersist 不需要)。
- 解析器可以执行特定的优化以更快地加载。

序列化和反序列化支持 XML 名字空间和映射 XML 名字空间到 CLR 名字空间的定义文件。Mapper 类允许定义任意的从 XML 名字空间和标签名字到 CLR 类的映射。

下面的表列出由 System.Windows.Serialization 名字空间展现的例子成员。

类	
CustomXmlDeserializer	反序列化器辅助的基础类。如果需要解析器的设计或编辑时间反序列化方向, 则应产生这个类的子类。
CustomXmlSerializer	序列化辅助的基础类。如果需要解析器的设计或编辑时间序列化方向, 则应产生这个类的子类。
DependencyPropertyContext	有关进行序列化的当前属性的上下文信息。XamlSerializer 负责在序列化对象上的任何属性之前将这个类的实例放在上下文栈中。
DPSerializationVisibilityAttribute	指定此属性或方法如设计器序列化器所见的可视性。
EntityContext	ObjectContext 和 PropertyContext 的抽象基础类。
LayoutAttribute	指定标签的 LayoutType 的属性。
Mapper	处理可扩展标记语义(XML)名字空间同一资源标识符(URI)和 Microsoft®.NET 名字空间类型之间的映射。

NamespaceMap	包含 Mapper 用于 xmlNameSpaceUri 和 Assembly、urtNamespace 所查找的内容之间的映射的信息。
ObjectContext	有关当前被序列化的对象的上下文信息。XamlSerializer 负责将此类的实例放在 SerializeObject 中的上下文栈上。
Parser	用来创建“Avalon”树的“Avalon”元素解析类。
ParserContext	提供 Parser 所需的所有上下文信息。
PropertyContext	有关被序列化的当前属性的上下文信息。XamlSerializer 负责在序列化对象上的任何属性之前将这个类的实例放在上下文栈中。
TextContext	有关被序列化的当前对象的上下文信息。XamlSerializer 负责将此类的实例放在 SerializeObject 中的上下文栈上。
XamlAttributeNode	XamlAttributeNode
XamlClrArrayPropertyNode	XamlClrArrayPropertyNode
XamlClrComplexPropertyNode	XamlClrComplexPropertyNode，它是指定为 CLR 对象的子对象的对象。
XamlClrEventNode	XamlClrEventNode，它是任何对象上的 CLR 事件。注意这可能是 DependencyObject，或任何其他对象类型。
XamlClrObjectNode	XamlClrObjectNode，它是除 DependencyObject 之外的任何类。
XamlClrPropertyNameode	XamlClrPropertyNameode，它是任何对象上的 CLR 属性。注意这可能是 DependencyObject，或任何其他对象类型。

XamlClrPropertyParseException	用于解析特定异常的解析器属性的异常类。
XamlComplexDependencyPropertyNode	XamlComplexDependencyPropertyNode，它是指定为 DependencyObject 的 XML 子元素的 DependencyProperty。
XamlComplexPropertySerializer	XamlComplexPropertySerializer 用于对不能序列化为 XML 属性的属性进行序列化。相反，将 ComplexPropertiy 序列化为元素内的标记。
XamlDefAttributeNode	XamlDefAttributeNode
XamlDefTagNode	XamlDefTagNode
XamlDependencyObjectSerializer	扩展 XamlSerializer 以包括(特定于元素的)动画。
XamlDependencyPropertyNode	XamlDependencyPropertyNode，它是 DependencyObject 上的正常的简单 DependencyProperty。
XamlDependencyPropertyParseException	用于解析特定异常的解析器动态属性的异常类。
XamlDesignerSerializerAttribute	指定给定类的序列化器类型名称。
XamlDocumentNode	XamlDocumentNode
XamlElementNode	XamlElementNode，它表示 DependencyObject。这和 CLR 对象不同，因为它们可以有与其关联的 Dependency 属性。
XamlEndAttributesNode	XamlEndAttributesNode
XamlEndClrArrayPropertyNode	XamlEndClrArrayPropertyNode
XamlEndClrComplexPropertyNode	XamlEndClrComplexPropertyNode
XamlEndClrObjectNode	XamlEndClrObjectNode

XamlEndComplexDependencyPropertyNode	XamlEndComplexDependencyPropertyNode
XamlEndDocumentNode	XamlEndDocumentNode
XamlEndElementNode	XamlEndElementNode
XamlEndIDictionaryPropertyNode	XamlEndIDictionaryPropertyNode
XamlEndIListPropertyNode	XamlEndIListPropertyNode
XamlEndResourceNode	XamlEndResourceNode
XamlEndStartElementNode	XamlEndStartElementNode
XamlIDictionaryPropertyNode	XamlIDictionaryPropertyNode
XamlIEnumeratorSerializer	序列化该属性所指向的节点集合。
XamlIListPropertyNode	XamlIListPropertyNode
XamlIListSerializer	序列化该属性指向的 IList。
XamlIncludeTagName	XamlIncludeTagName
XamlLanguageNode	XamlLanguageNode
XamlLiteralContentNode	XamlLiteralContentNode
XamlNode	衍生所有其他节点的基础节点。
XamlParseException	用于特定于解析器的异常的异常类。
XamlPIMappingNode	XamlPIMappingNode 映射 XML 名字空间到 CLR 名字空间和集合。
XamlResourceNode	XamlResourceNode
XamlRootSerializer	XamlSerializer 被用来持续存储逻辑树。
XamlRoutedEventArgs	XamlRoutedEventArgs
XamlSerializationCallbackException	用于特定于序列化回调的异常的异常类。
XamlSerializer	XamlSerializer 被用来持续存储逻辑树。
XamlSerializerBase	XamlSerializer 的基础类，提供通用辅助函数。

XamlTextNode	XamlTextNode
XamlTextNodeSerializer	对特殊情况下 TextNode 覆盖 XamlSerializer。
XamlXmlnsPropertyNode	XamlXmlnsPropertyNode
XmlAttribute	封装 DependencyObject 特定于 XML 的属性的类。
XmlDictionary	控制 XML 名字空间映射的字典。
XmlParserDefaults	由 Avalon 使用的公共类。

接口	
IAddChild	IAddChild 接口用于解析允许对象或文本出现在不直接映射到属性的标记中的标签之下的对象。
IBamlSerialize	写入 BAML 文件的动态属性的序列化接口。
ILoaded	此接口提供只读的称为 IsLoaded 的布尔属性，CLR 事件处理程序调用 Loaded、DeferLoad 和 EndDeferLoad 方法。
IPageConnector	提供由 BamlReader 对已编译的内容内部使用的方法。
IParseLiteralContent	用于 LiteralContent — 有其自身的解析器和保存器的内容。
IUriContext	IUriContext 接口向元素（如 Frame、PageViewer）和类型转换器（如 ImageDataTypeConverter）提供确保解析器在其上设置基础 URI，及 XAML、BAML 和 CAML 情况的代码生成的方法。

枚举	
LayoutType	可以和 Object 关联的布局类型。

SerializationAction	描述序列化器或反序列化器在它回调 CustomXamlSerializr 或 CustomXamlDeserializer 辅助类之后所采取的动作。
SerializationErrorAction	描述序列化器或反序列化器在报告错误时采取的动作。
XamlNodeType	XamlNode 的标识符。

### 互操作名字空间

互操作名字空间 330 提供一组支持和其他操作系统或计算平台进行互操作的类。下面的表列出由 System.Windows.Interop 名字空间展现的例子成员。

类	
ApplicationProxy	Application 上的 MarshalByRefObject 包装器，允许跨 AppDomain 和 Application 对象交互并允许在不同的线程上创建 Application。
DocobjHost	实现用于浏览器寄宿的 Docobj 服务器的管理的部分的互操作类。
PresentationInteropHelper	VisualInteropHelper
WindowInteropHelper	实现 Avalon WindowsInteropHelper 类，它有助于 B/W 遗留系统和 Avalon 窗口之间的互操作。

代理	
AppEntryPoint	应用代码入口方法的代理。

### Forms.Interop 名字空间

Forms.Interop 名字空间 332 提供允许应用寄宿表单控件操作的元素。

### System. IO. CompoundFile 名字空间

另一个名字空间，System. IO. CompoundFile(未在图 3 中示出)提供利用在其中存储各种文档可分发文件的复合文件的服务。这些服务用于内容的加密和压缩。这些服务也支持存储同一内容的多种表示，如可重设流的文档和固定格式的文档。下面的表列出由 System. IO. CompoundFile 名字空间展现的例子成员。

类	
CertificateHelper	取得签名数字证书的辅助类。
CompoundFileByteRangeReference	引用复合文件流内的字节范围的子流引用组件。
CompoundFileIndexReference	引用复合文件流内的逻辑入口的子流引用组件。
CompoundFileReference	对复合文件的一部分的引用。
CompoundFileReferenceCollection	复合文件引用的只读集合。
CompoundFileStorageReference	对容器存储的逻辑引用。
CompoundFileStreamReference	对容器流的逻辑引用。
CompressionTransform	用在复合文件数据空间中的压缩转换。
ContainerHelper	此类使得用户可以基于加载的 Application 实例获取对 StorageRoot 当前容器的访问。
DataSpaceManager	用于处理“Avalon”容器的特定实例内的数据空间。这就是数据转换模块插入容器 来支持类似的数据压缩和数据加密的方式。
DigitalSignature	支持客户检查和校验现有数字签名的只读类。
DigitalSignatureCollection	数字签名的只读集合。
DigitalSignatureManager	这个类用于在复合文件中创建、持续存

	储和处理数字签名。
DocumentSummaryInfo	包含对应于标准 OLE 文档摘要信息属性集合中的属性的属性元素。
FormatVersion	用于处理版本对象的类。
InstanceDataFormatException	在硬盘上 DrmTransformInstanceData 的格式非法时抛出的异常。
OleProperty	
PropSet	
RenditionInfo	处理表示信息的类。这个类也提供用于增加表示到存储与流及从中移除表示的 API。
RenditionInfoCollection	强类型的 RenditionInfo 对象集合。
RenditionManager	用于处理容器中的表示的类。
RightsManagementEncryptionTransform	实现 IDataTransform 接口的类，因此它可以用作实现内容的 RM 加密和解密的容器数据转换。
StorageInfo	用于处理容器文件中的存储的类。
StorageRoot	表示主容器类。每个容器文件有 StorageRoot 的一个实例。
StreamInfo	提供访问以处理容器文件中的流。
SummaryInfo	
TransformEnvironment	对每个转换对象给出此类的实例，作为转换对象和数据空间管理器提供的环境交互的方法。在给定的 TransformEnvironment 对象上保存转换对象的引用不是必须的，因为如果不需要和转换环境进行交互，它可能选择忽略。
TransformInitializationEventArgs	传递参数到事件处理程序中的公共类。

UseLicenseInfo	包含描述从 Tungsten 服务器返回的使用许可证的信息。
UseLicenseInfoEnumerator	表示存储在 DRM 转换的实例数据中的使用许可证的枚举器。
VersionTuple	包括主编号和次编号的版本元组类。
XmlDigitalSignatureProcessor	由根据 W3C 推荐对数据进行签名和校验的数字签名管理器使用的签名处理器。

接口	
IDataTransform	由所有数据转换对象实现的接口。
ILicenseStorage	此接口用于通过构建此接口的定制实现来分离 RMTransform 和 RMWizard，使得第三方能够利用 RMWizard 而不比强制他们使用 RMTransform。
ISignatureProcessor	签名处理器接口。
IUnknownRCW	此接口用于不透明地将任何(包装在运行时 COM 包装类中的)COM 接口作为 IUnknown 处理。

枚举	
DigitalSignatureProcessor	预定义的数字签名处理器。
SignatureProcessorNameType	签名处理器的类型。
TransformIdentifierTypes	当命名转换对象时，可以用几种方式中的一种解释传入的字符串。此枚举类型用于指定标识字符串的语义。

代理	
InvalidSignatureHandler	对散列表校验失败的签名进行调用。
TransformInitializeEventHandler	初始化转换的代理方法。

### System.Windows.Automation 名字空间

System.Windows.Automation 包含用于支持可访问性及用户接口自动化的成员。可访问性系统包括客户端和提供者端。工具的系统包括客户端自动化工具，客户端自动化工具包含用于搜索用户接口信息的客户自动化类。客户自动化类包括事件注册工具和逻辑元素发现工具。这组工具进一步包括用于向客户提供用户接口信息的提供者端自动化工具。提供者端自动化工具包括包含用于向客户提供事件信息的工具的自动化提供者类。

客户自动化类提供用于一个或多个客户的 UI 自动化方法。客户自动化类包含不特定于任何 UI 元素的方法。客户自动化类可以提供从点、窗口句柄或桌面根元素获取逻辑或原始元素的方法。客户自动化类还可以提供基于输入准则查找逻辑元素的方法。客户自动化类最好还包括注册和注销事件通知的方法。自动化类最好还包括用于加载代理 DLL、检索本地化的属性名称和控件模式并执行元素比较的辅助函数。客户自动化类也包括由客户用来监听事件的方法。下面的表列出由 System.Windows.Automation 名字空间展现的例子成员。

类	
ApplicationWindowPattern	展现通常和顶层应用窗口关联的行为和信息。客户可以使用这个类来平铺或层叠应用的多文档接口 (MDI) 子元素，在任务栏上找出其按钮，并定位用户接口众所周知的部分，如工具栏和菜单。
AssociatedInformationPattern	展现表示其他对象的 UI 元素的语义和元数据。
Automation	包含客户的用户接口 (UI) 自动化方法 (辅助技术或自动化测试脚本)。这些方法并不特定于特殊的元素。
AutomationEvent	请勿使用。这个类将在未来版本中去除。
AutomationEventArgs	模式或定制事件参数类。

AutomationFocusChangedEventArgs	焦点事件参数类。
AutomationIdentifier	对象身份的基类—基础标识符。这个类实际上是抽象的；仅实例化衍生的类。
AutomationIdentifierProxy	用于反序列化 AutomationIdentifier 的内部类。请勿直接使用。
AutomationPattern	请勿使用。这个类将在未来版本中去除。
AutomationPermission	提供一组访问 UI 元素的许可。不能从这个类继承。
AutomationPermissionAttribute	为 AutomationPermissionAttribute 提供方法和属性。不能从这个类继承。
AutomationProperty	请勿使用。这个类将在未来版本中去除。
AutomationPropertyChangedEventArgs	PropertyChanged 事件参数类。
AutomationTextAttribute	自动化文本属性的标识符。
AutomationTextPointer	表示字符在文本中的位置。AutomationTextPointer 提供访问文本和进行文本浏览的方法和属性。
AutomationTextRange	用于获取、设置、增加和移除选择。
AutomationTextSelection	目的：AutomationTextSelection 对象处理所有的文本选择管理。插入指针所在的选择是活跃的选择。例子用法：当客户希望增加、移除、修改或设置选择时使用。客户也可以通过 AutomationTextSelection 找出当前选择了什么。
AutomationTextUnit	自动化文本单位的标识符。
BasePattern	内部类。

DockPattern	展现元素在运行时改变其泊放状态的能力。
ElementNotEnabledException	这个异常在客户代码尝试处理当前未启用的元素或控件时抛出。
ElementPath	ElementPath 提供接下来返回原先由应用提供者记录、修改、或完全创建的逻辑元素所需的准则。
ExpandCollapsePattern	展现控件扩展以显示更多内容或折叠以隐藏内容的能力。例子包括菜单按钮、开始按钮、Windows 资源管理器中的树形视图项及组合框。
GridItemPattern	展现网格的元素。允许客户快速确定网格项的坐标。
GridPattern	展现级别网格功能、大小和浏览指定单元格。
HierarchyItemPattern	独立于它们在逻辑树中的关系展现控件的用户接口元件之间的层次关系。
Input	提供发送鼠标和键盘输入的方法。
InvokePattern	由执行单个明确的动作的对象(控件)实现。这些控件中多数不维护状态；调用它们触发应用中的动作。实现此接口的控件的例子包括按钮、工具栏按钮、菜单项、超级链接、复选框、单选按钮、树形视图控件中的加号及 Windows 资源管理器中的列表视图项。
LogicalElement	表示逻辑树中的 UI 元件而不管它的实现(“Avalon”、Microsoft® Win32®)。
LogicalStructureChangedEventArgs	逻辑结构改变事件参数类。
MultipleViewPattern	展现元素在同一组信息、数据或子元素的多种表示之间切换能力。例子包括

	ListView(缩略图、平铺、图标、列表、细节)、Excel 图表(饼图、直线图、直方图、带有公式的单元格的值)及 Outlook 日历(年、月、周、日)。
NoClickablePointException	当如由 GetClickablePoint 定义的参数值超出允许的范围值之外时抛出此异常。例如，当边界矩形为空，没有宽度或高度或该点上的 LogicalElement 和所调用的不同时。
ProxyAssemblyNotLoadedException	此异常在加载代理组合出现问题时抛出。这可能发生在以响应 Automation.RegisterProxyAssembly 或在加载缺省代理当碰到第一个 hwnd 基础 LogicalElement 时发生。
RangeValuePattern	展现控件管理有限范围内的值的能力。它传达控件的有效最小和最大值及当前值。
RawElement	表示原始元素树中的元素。
ScrollPattern	表示表达值的 UI 元素。
SelectionItemPattern	表示管理选择的容器内的单个项。UI 自动化客户使用这个类来得到有关支持 SelectionItemPattern 控件模式的 UI 元素的信息或对其进行处理。例子包括任何可以在支持 SelectionPattern 控件模式的控件中选择的项，如列表框或树形视图中的项。
SelectionPattern	表示管理选择的容器。由 UI 自动化客户使用以得到有关支持 SelectionItemPattern 控件模式的 UI 元素的信息或对其进行处理。
SortPattern	展现容器的当前排序顺序并允许客户用

	程序重排元素。
SplitPattern	表示可以通过创建邻接的窗口克隆它们自己的窗口。
TableItemPattern	表示有头部信息的网格项。
TablePattern	表示有头部信息的网格。
TextPattern	表示文本，如编辑控件。
TopLevelWindowEventArgs	TopLevelWindowEventArgs 事件参数类。
TreeLoadEventArgs	TreeLoadEventArgs 事件参数类。
ValuePattern	展现不跨范围的控件，如单选按钮、切换按钮、复选框、编辑框、RGB 颜色值和可选中的菜单项的值。
VisualInformationPattern	展现向用户传达信息的图形或动画有关的信息。这样的图形或动画如 IE 浏览器中指示正在下载文档的飘动旗帜或旋转地球或 Windows 资源管理器中指示正在进行复制的飞行纸片。
WindowPattern	展现元素改变其在屏幕上的位置或大小的能力。例子包括顶层应用窗口 (Word 或 Windows 资源管理器)、Outlook 的主窗口窗格 (目录、电子邮件消息、任务) 及 MDI 子窗口。
ZoomPattern	展现控件中的当前缩放水平并允许客户用程序改变它。

枚举	
AutomationPermissionFlag	包含 AutomationPermission 对象的访问标志。
ChildArrangement	安排子元素的不同方法。
DockPosition	可泊放窗口所附着的容器边缘。

ExpandCollapseState	由 ExpandCollapse 模式使用来指示扩展/收缩状态。
GetDataOptions	GetData 的选项。
HierarchyItemState	指示层次项状态的枚举，层次项状态包括：收缩、扩展或部分扩展。
ItemCheckState	指示复选框、单选按钮和类型控件状态的枚举。
LoadState	树状态标志。
LogicalMapping	指示是否在逻辑树中显示原始树中的节点的值。
ProxyFlags	用于指示请求属性的结果的枚举。
RowOrColumnMajor	表中的数据最好是由行表示还是由列表示。
ScopeFlags	用来定义何时监听事件的范围的标志。
ScrollAmount	由 ScrollPattern 使用来指示滚动方向(正向或反向)和滚动量(页或行)。
SendMouseInputFlags	SendMouseInput 的标志。这些标志指示是否发生了移动或是否按下或释放了按钮。
SplitStyles	分裂窗口的方向的标志。
StructureChangeType	指示逻辑树结构改变的标志。
TextPointerGravity	Gravity 确 定 当 在 AutomationTextPointer 插入文本时 AutomationTextPointer 的位置。Gravity 总是从文本指针的视角出发，即，我在我的字符前面或我在我的字符后面。当用户在文本指针的位置插入字符时，Before 意味着指针将放在新字符的前面；After 意味着指针将放在新字符的后面。

WindowChangeType	指示顶层窗口的改变的标志。
WindowInteractionState	用于用户交互的窗口的当前状态。
WindowVisualState	用来描述窗口可视状态的状态。 WindowVisualState 遵循 Office 和 HTML 对WindowState 的定义。

结构	
MatchCondition	指定使用 FindLogicalElement 或 FindRawElement 查找元素的准则。
NameAndData	包含作为对象的名字、数据及作为字符串的数据。
ProxyDescription	包含有关代理的信息的结构。
SortInformation	用来排序的信息。

代理	
AutomationEventHandler	处理 AutomationEvent 的代理。
AutomationFocusChangedEventHandler	处理焦点改变事件的代理。
AutomationPropertyChangedEventHandler	处理自动化属性改变事件的代理。
LogicalStructureChangedEventHandler	处理逻辑结构改变事件的代理。
ProxyFactoryCallback	由 HWND 处理程序实现；由 UI 自动化框架调用来请求指定窗口的代理。如果支持的话应返回代理，或在不支持的情况下返回空。
TopLevelWindowEventHandler	处理顶层窗口事件的代理。

### System.Windows.Ink 名字空间

System.Windows.Ink 名字空间提供支持电子墨水处理系统的类。电子墨

水处理技术对各种软件应用都很有用。这些电子墨水处理技术特别适合用于分析电子墨水，包括布局分析、分类和识别电子墨水。某些电子墨水处理技术允许电子墨水处理和实现这些技术的软件应用的操作异步，从而能够以不中断或不明显延迟软件应用操作的方式来处理电子墨水。软件应用甚至可以在正在处理前面的电子墨水输入时继续接受新的电子墨水输入。

应用编程接口实例化墨水分析器对象，墨水分析器对象从来自寄宿该文档并允许在第一个处理线程上的软件应用接收用于包含电子墨水内容的文档的数据。然后墨水分析器对象使用第一个线程做出文档数据的拷贝，向电子墨水分析处理提供文档数据的拷贝，然后将第一个处理线程的控制返回到分析处理。在分析处理分析了电子墨水之后，墨水分析器对象协调分析处理的结果和该文档的当前文档数据。

在特殊的实施例中，可以基于它们相对于彼此的空间位置对文件或文档中的元素进行描述。例如，可以用同样的空间坐标系统描述电子墨水笔触和键入的文本两者。使用空间信息来描述文档的元素，管理文档的软件应用可以维护描述其文档元素之间关系的数据结构。特别是，软件应用可以维护描述各种文档元素的类并定义各种文档元素之间的关联的数据接口。例如，这些关联可以定义为用于链接电子墨水笔触数据或它们的集合与电子文档中的其他元素(如单词、行、段落、图形、表格单元等等)的信息。

通过基于它们的空间位置来描述文件或文档数据结构中的文档元素，各种文件类型的文档元素可以使用通用技术来识别和处理它们的文档元素。更特别地，各种软件应用可以基于它们的空间位置描述文档内的文档元素，并通过使用通用电子墨水分析方法来使用此空间位置。进一步来说，通过指定文档中用于分析的特殊区域，每个软件应用可以限制分析处理为仅处理文档内所需的元素。

为了分析输入到文档中的新电子墨水，管理文档的软件应用修改和文档关联的数据结构以包括将分析的新墨水。然后软件应用向墨水分析工具提供此数据结构(或其相关部分)，墨水分析工具复制此数据结构的部分或全部用于分析(并操作数据的这个拷贝，而此数据独立于应用程序的文档数据结构)。墨水分析工具将该拷贝传送到分析处理，如解析处理(如，布局分析处理和/或分类处理)。软件应用可以在正在执行墨水分析处理时继续其正常操作，包括接收新的电子墨水输入和/或其他数据。除了接收新的电子墨水，应用程序

可以接收任何“其他数据”，例如，修改现有墨水、文本、图像、图形、表、流程图、图表等等的大小、位置或内容的数据；增加附加文本、图像、图像、表、流程图、图表等等的数据；删除现有文本、图像、图形、表、流程图、图表等等的数据。在所有需要的分析处理完成后，将分析结构返回到墨水分析工具。

墨水分析工具执行各种功能来帮助电子墨水的处理。墨水分析工具可以作为编程接口(如 API)来实现。进一步来说，墨水分析工具可以作为一组软件对象例程和相关信息来实现，它们可以由软件应用根据需要进行调用以分析文档中的墨水。

在一个实施例中，实现墨水分析工具的 API(此后称为墨水分析 API)可以包含两个核心类。第一个类被称作“分析上下文”类，而第二个类为“墨水分析器”类。“分析上下文”类的组件用于创建分析上下文对象。“墨水分析器”类的组件用于创建和使用对分析处理提供文档独立的分析上下文对象的对象、确定何时产生分析结果，并协调分析结果与文档的当前状态。

分析上下文类由宿主应用实现以创建分析上下文对象，它作为软件应用的内部文档树的代理视图来使用。分析上下文对象包含所有未分析的墨水数据，且分析上下文对象用于识别应分析哪些未分析的墨水数据。分析上下文对象也包含有关原先分析好的墨水的信息。原先分析好的墨水可以用于确定如何分析目前正在分析的墨水，且可以在分析未分析的墨水过程中对它自身进行修改。进一步来说，分析内容对象包含有关文档非墨水内容的信息，它用于正确地将墨水分类为非墨水内容的注释。

分析上下文类包括当由软件应用调用时创建分析上下文对象的构造函数。这个类也包括用于分析上下文对象的各种属性，包括称为“`DirtyRegion{get;}`”的属性。`DirtyRegion` 属性定义文档中包含未分析的墨水数据的部分(从而包括分析上下文对象的部分)。

分析上下文类也可以包括称为“`Rootnode{get;}`”的属性，它标识分析上下文对象中的最顶层或根上下文节点。这个根上下文节点，如子上下文节点一样，包括给定分析上下文对象的所有其他上下文节点对象。在特殊的实施例中，根上下文节点应属于上下文节点类型“Root”。在应用实现其自身的分析上下文对象的实施例中，分析上下文对象可以有其他上下文节点作为根上下文节点的兄弟节点，但墨水分析器类的组件只限于考虑由根上下文节

点包含的上下文节点。

分析上下文类可以附加地包括属性“`AnalysisHints{get;}`”，它返回由软件应用设置的分析提示对象数组。如下面更详细的说明，分析提示对象可以包含辅助分析处理的任何类型的信息。此信息可以包括，例如，仿真陈述、指南或单词列表。它也可以包括设置用于分析的语言的信息、将未分析的墨水作为完全的手写文本、完全的图画或对解析处理提供任何类型的指导来识别(如将墨水作为列表、表、形状、流程图、连接器、容器等等识别)的信息。

除了这些属性，分析上下文类也可以包括可以由例如软件应用调用从而让分析上下文对象执行任务的各种方法。例如，分析上下文类可以包括称为“`FindNode(Guid id)`”的方法。分析上下文对象中的每个节点都有全局唯一标识符(或 GUID)，且此方法将在分析上下文对象中的任何地方定位调用中指定的节点。

如分析上下文类那样，墨水分析器类也定义允许软件应用创建该类的实例(即，墨水分析器对象)的公共构造函数，和各种属性。例如，它可以包含称为“`UserInterface Context{get; set;}`”的属性，它定义向其返回分析处理结果的处理线程。此属性允许将结果同步到另一个对象。例如，如果这设置为主表单，那么解析器结果将在应用主线程发出。它也可以包含属性“`AnalysisOptions AnalysisOptions{get; set;}`”，此属性指定可以用于分析处理的各种准则。这些准则可以包括，例如，支持文本识别、支持表的使用、支持列表的使用、支持注释的使用和支持连接器和容器的使用。

墨水分析器类将包括各种方法。例如，这个类可以包含称为“`AnanysisRegion Analyze()`”的方法。此方法启动一个同步分析处理。传入文档元素数据到这个方法，它描述文档的当前状态并指示需要分析文档中的什么墨水。对某些实施例，如上所述，文档元素对象可以作为分析上下文对象提供(即，`AnalysisRegion Analyze(Analysis Context)`)。另外，可以将单个墨水笔触传递给分析处理，或者使用对笔触的引用(如 `AnalysisRegion Analyze (Strokes)`)或作为墨水分析器对象的属性引用(如，`InkAnalyzer.Strokes {get;set;}`)而没有属性传递给 `Analyze` 方法。

一旦分析处理完成，此方法将返回对已被修改为包含分析处理结果的独立于文档的分析上下文对象的引用。此方法也返回描述文档中计算出结果的区域的 `AnalysisRegion` 值。

墨水分析器类也可以包括称为“`AnalyzeRegion Analyze(AnalysisContext, waitRegion)`”的方法。此方法和上述的同步分析区域 `Analyze()` 方法相同，但是它只在在需要指定的 `waitRegion` 区域中的结果时对墨水进行分析。更特别地，对此方法的调用将标识文档的分析上下文对象和分析上下文对象的区域（称为“`waitRegion`”），分析处理应对此区域进行同步分析。在特定实施例中，将忽略分析上下文对象的所有其他区域，除非分析处理需要分析那些区域中的内容来完成它对 `waitRegion` 的分析。如上所述，用此方法传递的分析上下文对象包括称为“`DirtyRegion`”的属性，它描述文档中需要分析的区域。通过指定具体的 `waitRegion`，软件应用可以更加快速地获取所关注的一个特定区域的分析结果，而不是分析文档中的所有墨水数据。当调用这些 `Analyze` 方法中的任何一个时，将执行每个可用的分析处理。同时，由于这些 `Analyze` 方法是同步调用，不需要在它们结束时执行协调处理，也不需要在它们结束时触发事件。

墨水分析器类也可以包括称为“`BackgroundAnalyze(AnalysisContext)`”的方法。此方法启动指定的分析操作，但在独立的背景分析线程进行处理。因此，此方法将几乎立刻就返回控制到主处理线程，而实际的分析操作在背景完成。特别地，如果成功启动分析处理，此方法将返回“真”值。再次，传入该方法的 `AnalysisContext` 值标识文档的分析上下文对象并指示需要分析文档中的什么墨水。一旦分析操作已在背景线程上完成，则触发 `Result` 事件以允许软件应用访问这些结果。此事件包含结果和协调方法用于将结果返回包括在返回结果时文档的当前状态的分析上下文对象中。

这三个 `Analyze` 方法中的每一个都按顺序调用“`Analysis Region`”类中的方法“`Clone`”。使用“`Clone`”方法，这些 `Analyze` 方法创建独立的文档分析上下文对象，接下来此上下文对象由分析处理修改以展示分析结果。

墨水分析器类也可以包括称为“`Reconcile(AnalysisContext current, AnalysisResultsEventArgs resultArgs)`”的方法，软件应用在接收到由调用 `BackgroundAnalyze(AnalysisContext)` 方法导致的结果事件之后对它进行调用。`Reconcile` 方法 比较包含在文档独立的分析上下文对象中的分析结果与由软件应用维护的分析上下文对象的当前版本。此方法标识需要增加和从分析上下文对象的当前版本中移除的节点，并标识现有节点的任何下面的属性是否已改变：其识别结果、其位置、和该节点关联的墨水笔触，或和分析

操作的结果关联的任何其他数据。此方法也将这些标识出的改变写入到分析上下文对象的当前版本中。此方法对上下文节点顺序敏感，如行上下文节点上的单词上下文节点的顺序。

由于它们包含公共结果结构和私有结果结构，用此方法传回分析结果（即，`AnalysisResultsEventArgs` 属性的值）。返回公共结构从而软件应用可以预览将发生在协调阶段的改变。包括私有结构以防止软件应用在协调处理之前改变任何分析结果。

墨水分析器类也包括称为“`Recognizers RecognizersPriority()`”和“`SetHighestPriorityRecognizer(recognizer)`”的方法。当需要识别墨水时，将基于语言和能力使用适当的识别器。相应地，`Recognizers RecognizersPriority()`方法以墨水分析器对象评估它们的顺序返回识别处理。每个系统都取决于可用的识别处理来确定此顺序，但软件应用可以通过在墨水分析器对象上调用 `SetHighestPriorityRecognizer(recognizer)` 进行覆盖。墨水分析器将枚举这个有序列表直到可以找出适当的识别器。`SetHighestPriorityRecognizer(recognizer)` 方法提高识别处理的优先级。通过提高特殊的识别处理的优先级，如果它匹配当前识别操作所需的语言和能力，则将使用该识别处理。本质上，`SetHighestPriorityRecognizer(recognizer)` 将指定的识别处理压入到由 `RecognizersPriority` 方法返回的列表顶部。

墨水分析器类也包含称为“`AnalysisRegion Abort()`”的方法，它 can 将分析上下文对象用作参数。此方法允许前台或后台分析操作较早地终止。此方法返回描述在取消之前正在分析的区域的分析区域。因此，如果软件应用需要在晚些时候继续分析操作，可以将此区域合并到文档当前状态的分析上下文对象的 `DirtyRegion` 中。进一步来说，墨水分析器类可以包括称为“`AnalysisResultsEventHandler`”的事件，它根据需要的频率对墨水分析器对象触发。更特别地，可以在分析处理之间每隔五秒钟至少触发此事件一次。此事件可以用于向应用提供对正在进行的异步分析处理（或多个处理）的状态的更新。

墨水分析 API 也可以包括除分析上下文类和墨水分析器类之外的类。例如，墨水分析 API 可以包括上下文节点类。此类也可以包括和构成分析上下文对象和文档独立的分析上下文对象的上下文节点相关的各种组件，如称为

---

“`ContextNodeType Type{get;}`”的属性。每个上下文节点都有类型，且有每个类应遵循的一组特定规则。这包括这样的规则，如允许什么类型的子上下文节点，以及笔触是否可以直接和上下文节点关联或只能通过其子上下文节点关联。

上下文节点的可能类型可以在 `ContextNodeTypes` 枚举中定义，且可以包括，例如，下面的类型：`InkAnnotation` 节点，它表示墨水数据 注释非文本数据的；`InkDrawing` 节点，它表示构成图形的墨水数据；`InkWord` 节点，它表示构成单词的墨水数据；`Line` 节点，它表示 一个或多个 `InkWord` 节点和/或用于构成一行文本的单词的 `TextWord` 节点；`ListItem` 节点，它可以包含 `Paragraph`、`Image`，或类似的在列表中期望的节点；及 `List` 节点，它包括每个都描述列表中的一个条目的一个或多个 `Listem` 节点。节点类型也可以包括 `NonInkDrawing` 节点，它表示非墨水图形图像；`Object` 节点，它表示未由 `ContextNodeType` 枚举的其他值包括的数据；`Paragraph` 节点，它包含一个或多个对应于构成段落的行的 `Line` 节点；`Picture` 或 `Image` 节点，它们表示图像；`Root` 节点，它充当分析上下文对象中的顶层节点；`Table` 节点，它包含表示构成表的项目的节点；`TextBox` 节点，它表示文本框；`TextWord` 节点；及 `UnclassifiedInk` 节点，它对应于还未被分类的墨水数据。节点类型也可以包括用于其他节点分组的 `Group` 节点、用于着重表示项的 `InkBULLET` 节点、用于出现在表的一行中文档元素的 `Row` 节点，以及用于出现在表的单元格中的文档元素的 `Cell` 节点。

上下文节点也可以包括称为“`GUID Id {get;}`”的属性，它是当前上下文节点的全局唯一标识符。为了允许对任何所需上下文节点的访问，单个分析上下文对象内的每个上下文节点都应具有唯一标识符。此类也可以包括称为“`AnalysisRegion Location {get;}`”的属性，它标识文档空间中实际上定位相关上下文节点的位置。`AnalysisRegion` 是对一个或多个可能不相交的似矩形结构一起进行分组的二维结构。此类也可以包括称为“`StrokeCollection Strokes {gets;}`”的属性，它表示和相关的上下文节点关联的墨水笔触。对特殊的实施例，墨水分析 API 只允许叶上下文节点(如 `Word`、`Drawing` 和 `Bullet` 节点)有笔触。软件应用可以使用此属性通过所有祖先上下文节点来引用叶节点级别的笔触(如，根节点可以包含对相关的分析上下文对象中的所有笔触的笔触引用)。

进一步来说，这个类可以包括称为“`ContextNode ParentNode {get;}`”

的属性，它标识包含相关上下文节点的父上下文节点。在特殊的实施例中，上下文节点总是依赖于父上下文节点创建，而 Root 上下文节点为分析上下文对象的静态成员。这个类也可以包括属性“`ContextNode[] SubNodes {get;}`”，它标识作为相关上下文子节点直接子节点的所有上下文节点。即，此属性仅标识分析上下文对象中向下一层的那些子上下文节点。例如，`Paragraph` 上下文节点的这个属性值仅标识由该 `Paragraph` 节点包含的行上下文节点，而并不标识作为行上下文节点的子节点的单词上下文节点。

这个类也可以包括称为“`RecognitionResult RecognitionResult {get;}`”的属性，它提供由相关的识别分析处理或多个处理计算出的识别结果，因为 `RecognitionResult` 可以标识不止一种语言的不止一行文本。`RecognitionResult` 对文档独立的分析上下文对象中的每个上下文节点可用，虽然由识别分析处理设置并用于创建 `RecognitionResult` 对象的 `RecognitionData` 属性只可能在上下文节点树的一个层次上设置以避免复制数据。如果节点没有与其关联的 `RecognitionData`，它将或者合并其所有子节点的识别结果或从其父节点抽取识别结果。这个类也可以包括称为“`Stream RecognitionData {get; set;}`”的属性，它是 `RecognitionResult` 值的持续形式。再次，识别分析处理产生在相关上下文节点上设置的 `Stream RecognitionData` 值。然后基于此值构建 `RecognitionResult` 对象。

上下文节点类可以进一步包括称为“`ContextLink[] Links {get;}`”的属性，它提供 `ContextLink` 对象的数组。`ContextLink` 对象描述两个上下文节点之间的其他关系。虽然上下文节点通常和其他上下文节点有父子关系，`ContextLink` 支持上下文节点之间的其他关系。例如，`ContextLink` 可以支持两个上下文节点之间的连接：连接一个上下文节点到另一个上下文节点、由一个上下文节点包含另一个上下文节点，或由软件应用定义的所需类型的连接。可以通过调用 `AddLink` 方法将 `ContextLink` 增加到这个数组。类似地，可以通过调用 `DeleteLink` 方法从这个数组中移除 `ContextLink`。

进一步来说，这个类可以包括属性“`IsContainer {get;}`”和“`IsInkLeaf {get;}`”。如果相关上下文节点不是叶上下文节点(即，如果相关上下文节点包含子上下文节点并因此视为容器上下文节点)则 `IsContainer {get;}` 属性值为“真”，反之则为“否”。如果相关上下文节点是叶上下文节点则 `IsInkLeaf {get;}` 属性值为“真”，反之则为“否”。即，如果相关上下文节点不包含

任何子上下文节点，则将其视为叶上下文节点。在特定实施例中，期望 InkLeaf 上下文节点包含对笔触数据的引用，而容器上下文节点没有此限制。容器上下文节点可以引用也可以不引用笔触数据，根据软件应用的指派而定。

上下文节点类也可以包含属性“Rect RotatedBoundingBox {get; set;}”。由布局和分类分析处理来计算这个属性的值。如果以某个角度写出和相关上下文节点关联的墨水数据，则上下文节点边界将仍然是水平对齐的。然而 RotatedBoundingBox 属性的值将与写出和相关上下文节点关联的墨水数据的角度对齐。进一步来说，这个类可以包括属性“ReClassifiable {get;}”，此属性通知墨水分析器是否允许它修改相关上下文节点的值。

除了这些属性，上下文节点类也可以包含各种方法。例如，这个类可以包括称为“ContextNode CreateSubNode(ContextNodeType type)”的方法。这个方法允许创建特殊类型的子上下文节点。在一个实施例中，这个方法只允许创建相关上下文节点的合法子节点类型，从而防止创建异常的数据结构。例如，这个方法只允许 Line 上下文节点创建 InkWord 和 TextWord 子上下文节点。这个类也可以包含称为“void DeleteSubNote(ContextNode node)”的方法，它从相关分析上下文对象中删除引用的子上下文节点。然而，在某些实施例中，如果所引用的上下文节点仍然包含笔触或子上下文节点，则此方法会失败。如果引用上下文节点不是相关上下文节点的直接子节点，则此方法也会失败。如果软件应用实现其自身的分析上下文对象并随后使用此方法，则它不删除不是相关上下文节点直接子节点的一个或多个非空上下文节点，以防止在分析上下文对象中出现异常的数据结构。

另外，这个类也可以包括“ContextNode[] HitTestSubNodes(AnalysisRegion region)”方法，它返回定位在指定区域中的上下文节点的数组。然而，只返回此元素的直接子节点，而不是所有的后代节点。此区域由 AnalysisRegion 对象定义，它可以是一个或多个矩形的集合。在特殊的实施例中，如果上下文节点任何部分的位置和指定区域相交，则将在该数据中返回此上下文节点。此方法用于，例如，创建文档独立的上下文分析对象并协调分析结构与对应于文档当前状态的分析上下文对象。因此，频繁调用此方法，并应该为由墨水分析器对象快速重复访问而优化它。

上下文节点对象也可以包含称为“MoveStroke(Stroke stroke, ContextNode destination)”的方法。此方法将笔触关联从一个叶上下文节

点移动到另一个叶上下文节点。在某些实施例中，此方法仅在叶上下文节点之间使用。它也可以包含称为“MoveSubNodeToPosition(int OldIndex, int NewIndex)”的方法，此方法将相关上下文节点相对于其兄弟上下文节点重新定位。例如，如果文档有三个单词在一行中，如单词 1、单词 2 和单词 3，那么它们的顺序隐含在从父上下文节点返回的子节点数组中。此方法允许改变它们的顺序，从而相对于相关父上下文节点来说，通过将单词 1 的上下文节点从位置 1 移动到位置 3 可以把单词 1 指定为行中的最后一个单词。

进一步来说，这个类可以包含称为“AddLink(ContextLink link)”的方法，此方法增加新的 ContextLink 对象到当前的上下文节点。在特殊的实施例中，ContextLink 对象应包含对相关上下文节点的引用，以便将 ContextLink 成功增加到关联于相关上下文节点的 ContextLink 数组。它也包含称为“DeleteLink(ContextLink link)”的方法。此方法从相关上下文节点的 ContextLink 数组删除或异常指定的 ContextLink 对象。在一个实施例中，此方法调用总是成功完成，即使 ContextLink 不存在于关联于相关上下文节点的 ContextLink 数组中。

墨水分析 API 也可以包含分析提示类。如上述的很多类一样，分析提示类可以包括构造函数，称为“AnalysisHit()”，它初始化分析提示对象为空状态。这个类也可以包括多个属性，包括称为“AnalysisRegion Location {get;}”的属性。此属性指定文档中应用 AnalysisHint 的位置（作为 AnalysisRegion）。例如，如果文档是具有处于页眉的标题部分的自由格式笔记，那么应用可以对标题区域设置 AnalysisHint 来指定在该区域中期望水平的墨水行。AnalysisHint 将有助于提高分析处理的准确度。

此类也可以包含称为“string Factoid {get; set;}”的属性，它指定特殊的“仿真陈述”，用于文档中应用 AnalysisHint 的位置。仿真陈述向识别处理提供对墨水数据的期望使用（如，作为正常文本、数字、邮政编码、文件名和 Web URL）的提示。这个类也可以包括称为“RecognizerGuide Guide {get; set;}”和“OverrideLanguageId {get; set;}”的属性。RecognizerGuide Guide {get; set;} 属性指定应用于文档中应用 AnalysisHint 的位置的写入指导。写入指导可以，例如，通过向用户指定并通知识别器分析处理用户将在何处写出行或字符来帮助提高识别器分析处理的准确度。OverrideLanguageId {get; set;} 属性指定应用 AnalysisHint 的文档的语言提示。设置语言提示使得墨

---

水分析器对象使用指定的语言而不是上下文节点中指定的语言。

这个类也可以包含称为“PrefixText {get; set;}”的属性，它指定在将被识别的一行墨水之前写入或键入的文本。进一步来说，这个类可以包括称为“RecognitionModes RecognitionFlags {get; set;}”的属性，它指定识别处理在应用 AnalysisHint 的文档中的位置应遵循的特殊类型的模式。另外，这个类可以包括称为“SuffixText {get; set;}”的属性，它指定在将被识别的一行墨水之后写入或键入的文本，及称为“WordList WordList {get; set;}”的属性，它指定应由识别分析处理使用的单词的特殊集合。单词列表可以在用户实际上写入输入数据之前就知道期望的识别结果时使用，如期望在医药表单内写出的医药术语列表。

进一步来说，这个类可以包括称为“WordMode {get; set;}”的属性。如果此值为“真”，则分析处理将调整自身以偏向于对整个分析区域返回单个单词。它也可以包括称为“Coerce {get; set;}”的属性，它如果为“真”，则将强制分析处理将其结果限制在相关提示中设置的任何仿真陈述或单词列表值中。这个类也可以包括称为“AllowPartialDictionaryTerms {get; set;}”的属性。如果此属性的值为“真”，则将允许识别分析处理返回来自它的识别字典中的不完整单词。

在特殊实施例中，墨水分析 API 可以进一步包括 AnalysisRegion 类。这个类可以包括，例如，多个用于构造 AnalysisRegion 对象的构造函数。例如，它可以包含用于构造包含任何区域的 AnalysisRegion 对象的第一个构造函数，用于基于二维矩形的参数构造 AnalysisRegion 对象的第二个构造函数，及用于基于四个空间坐标构造 AnalysisRegion 对象的第三个构造函数。缺省的构造函数可以为，例如，创建空的区域。这个类也可以包含多个属性。例如，这个类可以包含称为“Rectangle Bounds {get;}”的属性，它检索 AnalysisRegion 的边界矩形；称为“IsEmpty {get;}”的属性，它指示相关的 AnalysisRegion 对象内部是否为空；及称为“IsInfinite {get;}”的属性，它指示相关的 AnalysisRegion 是否被设置为无限。

这个类也可以包含多个方法，如称为“AnalysisRegion Clone()”的方法，它克隆相关的 AnalysisRegion 对象。这个类也包括包含称为“Equals(AnalysisRegion otherRegion)”的方法，它检验指定的 AnalysisRegion 对象（称为 otherRegion）是否等同于相关的 AnalysisRegion

对象。如果指定的 AnalysisRegion 对象内部等同于相关的 AnalysisRegion 对象的内部，此方法返回“真”，否则返回“假”。

这个类可以进一步包含“Intersect(AnalysisRegion regionToIntersect)”方法，它剪切相关的 AnalysisRegion 对象为指定的分析区域。因此，结果 AnalysisRegion 对象将只包括和指定分析区域重叠或相交的区域。这个类也可以包含称为“Intersect(Rectangle rectangle)”的方法，它剪切相关的 AnalysisRegion 对象为指定的矩形。再次，结果 AnalysisRegion 对象将只包括和指定分析区域重叠或相交的区域。它也可以包括称为“MakeEmpty()”的方法，此方法把相关的 AnalysisRegion 对象内部初始化为空，及称为“MakeInfinite()”的方法，此方法把相关的 AnalysisRegion 占据的区域设置为无限。它可以进一步包括用于合并或分离不同的定义的区域的各种方法，如称为“Union(AnalysisRegion regionToUnion)”的方法，它指定合并或增加到相关 AnalysisRegion 对象的 AnalysisRegion 对象，和称为“Union(Rectangle rectangle)”的方法，它将指定的矩形合并到相关的 AnalysisRegion 对象。通过此方法，可以根据相关 AnalysisRegion 对象的坐标空间指定矩形。当然，这个类可以包含大量的其他方法，用于基于任何区域的所需定义合并区域或从另一个区域中抽取一个区域的。

墨水分析 API 也可以包括识别结果类。如上述的很多类，识别结果类可以包括一个或多个构造函数。例如，这个类可以包括称为“RecognitionResult(Stream lattice)”的构造函数，它通过给定的识别网格外构造识别结果对象。在特殊的实施例中，识别网格是识别处理结果序列化的形式。这个方法可以，例如，将识别网格指定为用于构建相关识别结果对象的字节数组。它也可以包括称为“RecognitionResult(ContextNode node)”的构造函数”，它通过给定的上下文节点构造识别结果对象。它也可以包括构造函数称为“RecognitionResult(string Text, int StrokeCount)”，它通过指定的文本值构造识别结果对象，它接下来关联于指定数量的笔触，且可以在识别处理没有对应于实际写出的墨水数据的替换识别值时用于修正。进一步来说，这个类可以包括称为“RecognitionResult(RecognitionResult leftRecognitionResult, RecognitionResult rightRecognitionResult)”的构造函数，它通过合并两

个现有的识别结果对象来构造识别结果对象。

识别结果类也可以包含一个或多个属性，如称为“`StrokeCollection StrokeCollection {get;}`”的属性，它提供标识包含在单个墨水对象中的笔触集合的笔触索引数组，及称为“`RecognitionAlternate TopAlternate {get;}`”的属性，它提供识别结果的最佳替换。这个类也可以包括称为“`RecognitionConfidence RecognitionConfidence {get;}`”的属性，它提供来自识别分析处理的当前结果的顶层替换选择的置信水平(如，强、中等或弱)，及称为“`string TopString {get;}`”的属性，它返回来自识别分析处理的分析结果的最佳结果字符串。

识别结果类也可以包括多个方法，如称为“`public RecognitionAlternateCollection GetAlternateCollectionFromSelection (selectionStart, selectionLength, maximumAlternates)`”的方法，它指定来自识别分析处理的分析结果的最佳结果字符串内的选择的替换集合，其中每个替换只对应于一个墨水片段。此方法的输入参数可以包括，例如，指定从中返回替换集合的文本选择开始位置的值、指定从中返回替换集合的文本选择长度的值，及指定返回替换的最大数量的值。此方法可以从识别结果的最佳结果字符串内的选择中返回 `RecognitionAlternateCollection` 替换集合。

识别结果类可以进一步包括称为“`RecognitionResult Merge(RecognitionResult left, string separator, RecognitionResult right)`”的方法。此方法可以用来通过单个字符串创建新的识别结果对象，得到扁平的网格，附加单个字符串到现有识别结果对象开始或末尾，或在两个现有的识别结果对象之间连接单个字符串。这个类也可以包括称为“`ModifyTopAlternate(RecognitionAlternate alternate)`”的方法，它指定用已知的替换来修改识别结果。对某些实施例，缺省的识别分析处理的结果最佳结果字符串对应于顶层替换。然而，此方法可以用于指定将除顶层替换之外的替换用在识别分析处理结果中。如果新的顶层替换导致和原先不同的划分，`ModifyTopAlternate` 方法将自动更新上下文节点来反映这些改变。为了检索可以用来修改识别结果的替换，此方法调用 `GetAlternatesFromSelection` 方法。这个类也可以包括称为“`Stream Save()`”的方法，它持续地维护形式为识别网格的相关识别结果对象。识别网格是用

于表达识别处理结果的序列化的格式。

墨水分析 API 也可以包括 AnalysisOptions 枚举类型。此类型可以包含一个或多个指定分析处理如何分析墨水数据的字段，如称为“const AnalysisOptions Default”的字段，它支持分析处理的所有可用选项。此字段可以，例如，支持文本识别、表的使用、列表的使用、注释的使用、连接器和容器的使用，及中间结果的使用。此类型也可以包括称为“const AnalysisOptions EnableAnnotations”的字段，它启用和禁用注释的检测；称为“const AnalysisOptions EnableConnectorsAndContainers”的字段，它启用和禁用连接器和容器的检测；及称为“const AnalysisOptions EnableIntermediateResults”的字段，它启用和禁用返回使用不同的、顺序分析处理之间（如，解析处理和接下来的识别处理之间）的分析结果到软件应用。此类型也可以包括称为“const AnalysisOptions EnableLists”的字段，它启用和禁用列表的检测；及称为“const AnalysisOptions EnableTables”的字段，它启用和禁用表的检测。这个枚举类型进一步包括称为“const AnalysisOptions EnableTextRecognition”的字段，它启用和禁用文本识别分析处理。然而，如果附加的分析处理可用（或同一分析处理的不同版本可用），则此类型可以相应地包括附加的 AnalysisOption。

进一步来说，墨水分析 API 可以包括 AnalysisResultsEventArgs 类。这个类可以有称为“public AnalysisResultsEventArgs()”的构造函数，它创建包含分析结果的数据结构并在触发 AnalysisResults 事件时返回到软件应用。这个类也可以包括称为“InkAnalyzer InkAnalyzer {get;}”的属性，它标识执行分析处理的 InkAnalyzer 对象。

API 也可以包括 Line 类，它对将“Line”对象的使用识别为表示几何直线的某些类型的操作系统有用。这个类可以包括构造函数，如称为“public Line(Point beginPoint, Point endPoint)”的构造函数，它创建 Line 对象。这个类也可以包括各种属性，如称为“public Point BeginPoint {get; set;}”的属性，它表示行对象的起点及称为“public Point EndPoint {get; set;}”的属性，它表示行对象的终点。

除了这些类，墨水分析 API 也可以包含 RecognitionAlternate 类。这个类可以包括表示墨水片段和识别器的字典比较产生的可能单词匹配的元素。例如，这个类可以包括称为“Line Ascender {get;}”的属性，它提供存在

于单个行(表示为两个点的行)上的 RecognitionAlternate 对象的上升行; 称为“public Line Baseline {get;}”的属性, 它提供存在于单个行上的 RecognitionAlternate 对象的基线; 和称为“Line Descender {get;}”的属性, 它提供存在于单个行上的 RecognitionAlternate 对象的下降行。这个类也可以包括称为“RecognitionResult Extract {get;}”的属性, 它提供当前的 RecognitionAlternate 对象的 RecognitionResult 对象。这个属性可以用于, 例如, 从包含该单词的行的 RecognitionResult 对象抽取单词的 RecognitionResult 对象。

它也可以包括称为“Line Midline {get;}”的属性, 它提供存在于单个行上的行中点 RecognitionAlternate 对象; 称为“StrokeCollection Strokes {get;}”的属性, 它提供包含在墨水对象中的笔触集合(即, 它提供表示关联于 RecognitionResult 的笔触的 StrokeCollection), 及称为“StrokeCollection[] StrokeArray {get;}”的属性, 它提供包含在一个或多个墨水对象中的表示和 RecognitionResult 关联的笔触的笔触集合。这个类也可以包括称为“RecognitionConfidence RecognitionConfidence {get;}”的属性, 它提供识别分析处理在识别 RecognitionConfidence 对象或姿势时确定的置信水平(如, 强、中等, 或弱)。对非行节点, 将返回最低的相关上下文节点的子节点 RecognitionConfidence。它也可以包含称为“string RecognizedString {get;}”的属性, 它指定替换的结果字符串。因此, 对单词上下文节点之上的任何上下文节点, 通过此方法将结果字符串连接在一起。例如, 行节点将包含结果字符串, 结果字符串又包含所有其子节点或单词节点的结果。段落节点将包含结果字符串, 结果字符串又包含所有其子节点或行节点的结果

RecognitionAlternate 类也可以包含一个或多个方法, 包括例如, 称为“StrokeCollection[] GetStrokesArrayFromTextRange(int selectionstart, int selectionlength)”的方法, 它通过对对于已知文本范围的每个墨水对象指定 StrokeCollection。这个类也可以包含称为“StrokeCollection[] GetStrokesFromStrokesArrayRanges(StrokeCollection[] strokesArray)”的方法, 它指定包含已知输入笔触集合且识别器向其提供替换的最小笔触集合。更特别地, 通过每个均包含该集合的笔触索引数组的墨水笔触的数组返回笔触。通过此方法返回的墨水笔触集合可以和输入集合匹配, 或如果输入

集合仅匹配包括所有输入笔触的最小的识别结果的部分，那么它可以更大。这个类可以进一步包括称为“`StrokeCollection GetStrokeFromStrokesRanges(StrokeCollection strokes)`”的方法，它指定包含已知输入笔触集合且识别器可以对其提供替换的最小笔触集合；及称为“`StrokeCollection GetStrokesFromTextRange(int selectionstart, int selectionlength)`”方法，它指定对应于已知文本范围的`StrokeCollection`。

这个类可以进一步包括称为“`void GetTextRangeFromStrokes(ref int selectionstart, ref int selectionend, StrokeCollection strokes)`”的方法，它指定识别器可以对其返回包含已知笔触集合的替换的已识别文本的最小范围；及称为“`void GetTextRangeFromStrokesArray(ref int selectionstart, ref int selectionend, StrokeCollection[] strokesarray)`”的方法，它指定识别器可以对其返回包含已知笔触集合的替换的已识别文本的最小范围。它也可以包括称为“`RecognitionAlternateCollection SplitWithConstantPropertyValue(GUID propertyType)`”的方法，它返回替换的集合，它们是对其调用此方法的替换的划分。集合中的每个替换都包含对传递到该方法中的属性有相同属性值的邻接识别片段。例如，此方法可以用于获取根据识别结果中的置信水平边界（强、中等，或弱）、行边界，或片段边界划分初始替换的替换。它可以进一步包括称为“`byte[] GetPropertyValue(GUID propertyType)`”的方法，它指定替换中的已知属性值，如替换中的识别器置信水平。然而，不是所有的识别分析处理都会提供所有的属性类型的值。因此，此方法提供由相关的识别分析处理支持的类型的数据。

墨水分析 API 也包括`RecognitionAlternateCollection`类。如上述的其他类，这个类可以包括用于创建`RecognitionAlternateCollection`对象的称为“`RecognitionAlternateCollection()`”的构造函数。这个类也可以包括多个属性，如称为“`Count {get;}`”的属性，它提供包含在替换识别值的集合中的对象或集合的数目；称为“`IsSynchronized {get;}`”的属性，它提供指示对替换识别值集合的访问是否和软件应用同步（即，“线程安全”）的值；及称为“`SyncRoot {get;}`”的属性，它提供可用于同步对替换识别值集合的访问的对象。

这个类也可以包含一个或多个方法，如称为“`virtual void CopyTo(Array`

array, int index)” 的方法，它复制当前替换识别值集合中的所有元素到指定的一维数组，开始于指定的目标数组下标；及称为“`IEnumerable.IEnumerable.GetEnumerator()`”的方法，它是 `IEnumerable` 的标准实现，`IEnumerable` 使得调用者能够使用 `for each` 结构来枚举替换识别值集合中的每个 `RecognitionAlternate`。这个类也可以包括称为“`RecognitionAlternateCollectionEnumerator GetEnumerator()`”的方法，它返回包含识别替换值集合内的所有对象的 `RecognitionAlternateCollectionEnumerator`。此方法可用于，例如，检索识别替换值集合中的每个对象。

墨水分析 API 可以附加地包括 `RecognitionConfidence` 枚举和 `RecognitionMode` 枚举，它们都可以包含一个或多个和识别分析处理相关的字段。例如，`RecognitionConfidence` 类可以包含多个字段，如称为“`Intermediate`”的字段，它指示识别分析处理确信正确结果在提供的替换识别值列表中；称为“`Poor`”的字段，它指示识别分析不确信结果在提供的替换识别值列表中；及称为“`Strong`”的字段，它指示识别分析处理确信替换识别值中的最佳替换是正确的。

类似地，`RecognitionMode` 类可以包括指定识别分析处理如何解释电子墨水数据并从而确定识别结果字符串的字段。例如，这个类可以包括称为“`Coerce`”的字段，它指定识别分析处理基于对上下文指定的仿真陈述强制产生识别结果；及称为“`Line`”的字段，它指定识别分析处理把电子墨水数据当作单个行。这个类也可以包括称为“`None`”的字段，它指定识别分析处理不应用识别模式；及称为“`Segment`”的字段，它指定识别分析处理将电子墨水数据视为由单个单词或字符构成。进一步来说，这个类可以包括称为“`TopInkBreaksOnly`”的字段，它禁用多个分段。

进一步来说，墨水分析 API 可以包括 `ContextLink` 类，它定义如何将两个上下文节点链接在一起。`ContextLink` 对象自身表示链接哪两个上下文节点、链接的方向及链接的类型。这个类可以包括称为“`ContextNode SourceNode {get;}`”的属性，它指定链接到另一个上下文节点的源上下文节点；称为“`ContextLinkType LinkType {get;}`”的属性，它指定存在于源和目标上下文节点之间的链接关系的类型；及称为“`CustomLinkType {get;}`”的属性，它指定使用定制的链接。此情况在应用决定使用墨水分析器 API 的链接系统

来表示 API 可以识别的范围之外的特定于应用的链接时会发生。

这个类也可以包括称为 “ContextNode DestinationNode {get;}” 的属性，它指定链接到另一个上下文节点的目标上下文节点。可能有两个构造函数对这个类可用，它们创建现有的源和目标上下文节点之间的关系。

这个类也可以包括称为 “ContextLinkType enum” 的枚举，它定义由两个上下文节点共享的关系的类型。这些链接类型可以包括，例如，“AnchorsTo” 类型，它描述一个节点锚接到另一个节点。基于解析情况，两个节点都可以使用 SourceNode 或 DestinationNode 属性。链接类型也可以包括 “Contains” 类型，这描述一个节点包含另一个节点。具备此关系，容器节点可以作为 SourceNode 引用，而被包含的节点可以作为 DestinationNode 引用。链接类型可以进一步包括 “PointsTo” 类型，它描述一个节点指向另一个节点。对此关系，进行指向的节点可以作为 SourceNode 引用，而被指向的节点可以作为 DestinationNode 引用。进一步来说，链接类型可以由 “PointsFrom” 类型，它描述从另一个节点指向一个节点。在此关系，从其他节点进行指向的节点可以作为 SourceNode 引用，而被从其他节点指向的节点可以作为 DestinationNode 引用。

链接类型可以附加地包括 “SpansHorizontally” 类型，它描述一个节点跨越另一个节点的横向长度；及 “SpansVertically” 类型，它描述一个节点跨越另一个节点的纵向长度。对这些类型，覆盖(划出、强调、勾勒)其他节点的节点通常最后写出，并可以作为 SourceNode 引用，而被跨越的节点可以作为 DestinationNode 引用。链接类型也可以包括 “Custom” 类型，它描述使用了定制的链接类型。当使用此值时，ContextLink 对象上的 “CustomLinkType” 属性可以对此链接的目的提供更多细节。

## 应用模型

窗口客户机集成了 Web 的特性和传统桌面应用的特性。应用模型提供安全应用的框架并简化客户机应用的开发、部署和维护。此框架提供简单和一致的用户体验。例如，本地应用可以利用熟悉的类似浏览器的特性，而不管应用是寄宿在浏览器中还是单独的应用，同时保留在本地客户机上执行的好处。此框架允许用户利用他们熟悉 Web 这一点，从而增加了用户的舒适水平并减少了学习使用新应用所需的时间。应用模型是 System.Windows 名字空间

的一部分。

使用应用模型的应用以类似于 Web 页面的方式工作。当用户浏览应用时，自动地安装应用而无需要求用户对安装进行确认、重启客户机系统或承担其他应用出错的风险。在一个实施例中，逐步下载应用，从而在完全下载该应用之前提供初始水平的交互性。应用的更新自动地进行并以对用户透明的方式进行。因此，用户总是能够访问应用的最新版本而无需明确地执行应用升级。

使用应用模型的应用在客户机系统上本地运行，而不管客户机系统是在线(即，活跃地连接到 Web)还是离线(即，没有活跃地连接到 Web)。这允许应用提供比基于服务器的应用更好的性能，基于服务器的应用需要活跃的 Web 链接并通过 Web 和服务器持续地交互数据。在把应用安装到客户机系统上之后，可以通过“开始”菜单(象传统的桌面应用那样)或通过浏览到该应用(象 Web 应用那样)来访问应用。应用模型包括三个主要部分：应用生命周期管理、应用框架和浏览框架。

应用模型支持两种不同类型的应用：“在线应用”和“管理的应用”。使用应用模型的应用可以在浏览器或独立的顶层窗口中执行。“在线应用”是通过服务器执行并寄宿在服务器中的应用。可以缓冲此应用以进行离线访问或此需要特定的在线资源才能正确执行应用。“管理的应用”是离线提供并安装在客户机上的。操作系统对管理的应用提供服务。可以将管理的应用的入口添加到客户机上的“开始”菜单。可以逐步下载应用，以允许用户在正在下载应用时开始和应用交互，而不是延迟交互直到安装处理完成。

应用具有关联的应用清单，它描述应用的依存关系，如执行应用所需的附加库和资源。安装程序使用应用清单来控制应用的下载和安装。作为安装处理的部分来调用“信任管理器”。信任管理器使用应用清单来确定执行应用需要何种许可。应用清单也指定外壳信息，如文件关联及是否添加入口到开始菜单，以及该入口的图标和文本。

使用应用模型的应用包括标记、代码、资源和清单。应用由其应用对象定义和限定，应用对象是每次应用会话期间持续存储在存储器中全局对象。应用对象知道所有属于应用的资源并提供其自身与其他应用或外部资源之间的边界。应用框架使用应用对象来识别、引用应用，并和应用通讯。应用对象也在应用内使用，以管理窗口和资源、指定启动和关闭行为、处理配置设

置、指定应用的可视样式、在浏览之间共享代码、状态和资源，及处理应用范围的事件。

浏览框架支持基于浏览的应用，此类利用用户熟悉浏览操作这一点并将 Web 上的活动写入日志，以在客户机系统上提供更加熟悉、一致的用户体验，而不管应用是寄宿在系统浏览器中或在独立的顶层窗口中。写日志是由浏览框架用来追踪浏览历史的处理。日志允许用户以线性浏览序列回扫他们向后和向前的步骤。无论浏览体验是寄宿在浏览器中还是在独立的浏览窗口中，每次浏览都持续存储在日志中，并可以通过使用“向前”和“向后”按钮或通过调用“向前”和“向后”方法以线性序列重新访问。每个浏览窗口都有相关联的日志。

`NavigationApplication` 类通过提供和浏览相关的属性和事件简化创建基于浏览的应用的任务。`NavigationApplication` 类包括指定在应用最初启动时系统所浏览的页面或元素的启动属性。这个类也包括允许应用开发者跨页面共享全局状态信息而无需做出应用的子类，并支持到这些属性的数据绑定的属性集合。

#### 示例计算系统和环境

图 4 展示在其中可以(或者完整或者部分)实现编程框架 132 的适合的计算环境 400 的例子。计算环境 400 可以在这里所述的计算机和网络架构中使用。

例子计算环境 400 只是计算环境的一个例子且并不意味着对该计算机和网络架构的使用范围或功能有任何限制。也不应将计算环境 400 解释为对任何一个示例计算环境 400 中所示的组件或其组合有任何依存关系或需求。

可以用大量其他通用或专用计算系统环境或配置来实现框架 132。适合使用的众所周知的计算系统、环境和/或配置的例子包括，但不仅限于，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、小型机、大型机、包括上述任何系统或设备的分布式计算环境等等。该框架的紧凑版本或子集可以在有限资源客户机，如手机、个人数字助理、手持计算机，或其他通讯/计算设备上实现。

框架 132 可以在计算机可执行指令的通用上下文中说明，如由一个或多个计算机或其他设备执行的程序模块。通常，程序模块包括执行特殊任务或

实现特殊抽象数据类型的例程、程序、对象、组件、数据结构等等。框架 132 也可以实现在分布式计算环境中，在其中任务由通过通讯网络连接的远程处理设备执行。在分布式计算环境中，程序模块可以位于本地和远程计算机存储器存储媒体，包括存储器存储设备。

计算环境 400 包括形式为计算机 402 的通用计算设备。计算机 402 的组件可以包括，但不仅限于，一个或多个处理器或处理单元 404、系统存储器 406 及连接包括处理器 404 的各种系统组件到系统存储器 406 的系统总线 408。

系统总线 408 表示几种类型的总线结构中的任何一种，包括存储器总线或存储器控制器、外围设备总线，及使用多种总线结构中的任何一种的处理器或本地总线。作为例子，这样的结构包括工业标准结构 (ISA) 总线、微通道结构 (MCA) 总线、扩展的 ISA (EISA) 总线、视频电子标准协会 (VESA) 总线，及亦称为中层楼总线的周边元件互连 (PCI) 总线。

计算机 402 通常包括各种计算机可读媒体。这样的媒体可以为可由计算机 402 访问任何可用媒体并且包括易失和非易失媒体、可移动和不可移动媒体。

系统存储器 406 包括形式为易失存储器 (如随机访问存储器 (RAM) 410) 和/或非易失存储器 (如只读存储器 (ROM) 412) 的计算机可读媒体。包含基本的例程以例如在启动过程中帮助在计算机 402 的元件之间传输信息的基本输入/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 立即访问和/或当前正在操作的数据和/或程序模块。

计算机 402 也可以包括其他可移动/不可移动、易失/非易失计算机存储媒体。作为例子，图 4 展示了用于读写不可移动、非易失磁媒体 (未示出) 的硬盘驱动器 416，用于读写可移动、非易失磁盘 420 (如，“软盘”) 的磁盘驱动器 418，及用于读和/或写可移动、非易失光盘 424 (如 CD-ROM、DVD-ROM 或其他光学媒体) 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 中的每个都通过一个或多个数据媒体接口 426 连接到系统总线 408。作为选择，硬盘驱动器 416、磁盘驱动器 418 和光盘驱动器 422 可以通过一个或多个接口 (未示出) 连接到系统总线 408。

磁盘驱动器及与它们关联的计算机可读媒体向计算机 402 提供对计算机可读指令、数据结构、程序模块和其他数据的非易失存储。虽然上述例子展示了硬盘 416、可移动磁盘 420 和可移动光盘 424，应理解，也可以使用其他

类型的可以存储可由计算机访问的数据的计算机可读媒体，如盒式磁带或其他磁存储设备、闪存卡、CD-ROM、数字多用途盘(DVD)或其他光学存储、随机访问存储器(RAM)、只读存储器(ROM)、电子可擦可编程序只读存储器(EEPROM)等等来实现示例计算系统和环境。

可以在硬盘416、磁盘420、光盘424、ROM412和/或RAM410上存储任何数量的程序模块，这包括例如操作系统426、一个或多个应用程序428、其他程序模块430和程序数据432。操作系统426、一个或多个应用程序428、其他程序模块430和程序数据432(或其组合)中的每一个都可以包含编程框架132的元素。

用户可以通过如键盘434和定点设备436(如，“鼠标”)这样的输入设备输入命令和信息到计算机402中。其他输入设备438(未具体示出)包括麦克风、操纵杆、游戏垫、圆盘式卫星电视天线、串行口、扫描仪和/或类似设备。这些和其他输入设备通常通过连接到系统总线408的输入/输出接口440连接到处理单元404，但也可以通过其他接口和总线结构，如并行口、游戏口，或通用串行总线(USB)来连接。

显示设备442或其他显示设备也通过接口，如视频适配器444连接到系统总线408。除显示器442外，其他外围输入设备包括可以通过输入/输出接口440连接到计算机402的组件，如扬声器(未示出)和打印机。

计算机402可以使用到一个或多个远程计算机(如远程计算设备448)的逻辑连接在连网环境中运行。作为例子，远程计算设备448可以是个人计算机、可移动计算机、服务器、路由器、网络PC、对等设备或其他普通网络节点等等。远程计算设备448在图示中是包括在此相对于计算机402所述的很多或所有元件和特性的可移动计算机。

计算机402和远程计算机448之间的逻辑连接包括局域网(LAN)450和广域网(WAN)452。这样的连网环境在办公室范围或企业范围的计算机网络、内联网和因特网中是很常见的。

当用在LAN连网环境中时，计算机402通过网络接口或适配器454与局域网450相连。当用在WAN连环境中时，计算机402通常包括调制解调器456或其他在广域网452上建立通讯的方法。调制解调器456，可以为内置的或外置的，通过输入/输出接口440或其他适当机制连接到系统总线408。应理解所示的网络连接是示范性的且可以使用在计算机402和448之间建立通讯连

接的其他方法。

在连网的环境中，如计算环境 400 所示，相对于计算机 402 说明的程序模块或其部分可以存储在远程存储器存储设备中。作为例子，远程应用程序 458 驻留在远程计算机 448 的存储器设备中。为了说明，应用程序和其他可执行程序组件，如操作系统，在此展示为离散的块，虽然应承认这样的程序和组件以不同的次数驻留在计算设备 402 中不同的存储组件内，并由计算机的数据处理器执行。

框架 132 的实施例，更特别地，API 142 或对 API 142 做出的调用可以存储在某种形式的计算机可读媒体上或通过它来发送。计算机可读媒体是可由计算机访问的任何可用媒体。作为例子，而非限制，计算机可读媒体可以包括“计算机存储媒体”和“通讯媒体”。“计算机存储媒体”包括以任何存储信息(如计算机可读指令、数据结构或其他数据)的方法或技术实现的易失和非易失、可移动和不可移动媒体。计算机存储媒体包括，但不仅限于，RAM、ROM、EEPROM、闪存或其他存储器技术、CD-ROM、数字多用途盘(DVD)或其他光学存储、盒式磁带、磁带、磁盘存储或其他磁存储设备，或可以用来存储所需信息并可以由计算机访问的任何其他媒体。

“通讯媒体”通常以调制的数据信号(如载波或其他传输机制)实现计算机可读指令、数据结构、程序模块或其他数据。通讯媒体也包括任何信息发送媒体。术语“调制的数据信号”指设置或改变它的一个或多个属性从而在该信号中编码信息的信号。作为例子，而非限制，通讯媒体包括有线媒体(如有线网络或直接连接)及无线媒体(如声音、无线电、红外线和其他无线媒体)。上述任何媒体的组合也包括在计算机可读媒体的范围之中。

作为选择，框架的部分可以用硬件或硬件、软件和/或固件的组合来实现。例如，可以设计或编程一个或多个专用集成电路(ASIC)或可编程逻辑设备(PLD)来实现该框架的一个或多个部分。

概念上来说，编程接口可以一般地视为如图 5 或图 6 所示的那样。图 5 展示接口 I<sub>1</sub>，第一个和第二个代码段将其作为通讯的管道。图 6 展示由接口对象 I<sub>1</sub> 和 I<sub>2</sub>(它可以是第一个和第二个代码段的部分，也可以不是)组成的接口，接口对象 I<sub>1</sub> 和 I<sub>2</sub> 使得系统的第一个和第二个代码段能够通过媒体 M 通讯。在图 6 的视图中，可以将接口对象 I<sub>1</sub> 和 I<sub>2</sub> 视为同一系统的独立接口，也可以视为对象 I<sub>1</sub> 和 I<sub>2</sub> 加上媒体 M 构成了接口。虽然图 5 和图 6 展示了在流的

每一边都有双向的流和接口，特定的实现可以只包括一个方向上的信息流（或没有如下所述的信息流），或可以在一边只有一个接口对象。作为例子，而非限制，如编程接口或程序接口（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）接口这样的术语都包含在编程接口的定义之内，

这样的编程接口的各方面可以包括第一个代码段发送信息（其中“信息”以其最广泛的含义使用并包括数据、命令、请求等等）到第二个代码段的方法；第二个代码段接收信息的方法；及信息的结构、序列、语法、组织、方案、时间和内容。在这点上，只要以由接口定义的方式传输信息，下层传输媒体自身对接口的操作就不重要，无论是有线或无线媒体，还是两者的组合。在特定的情况下，当信息传输可以通过其他机制传输进行（如，将信息存放在独立于代码段之间信息流的缓冲、文件内等等）或不存在，如一个代码段简单地访问由第二个代码段执行的功能时，信息可以不在常规意义上的一个或两个方向上传输。在给定的情况下，任何或所有这些方面都可能很重要，如取决于代码段是否是松散连接或紧密连接配置的系统的一部分，且因此这个列表应被视为是说明性而非限制性的。

编程接口的概念为熟悉技术的人所知，并从上述对本发明的详细说明中可以清楚地理解。然而，存在实现编程接口的其他方法，并且除非明确排除，它们也将包括在说明书后面的权利要求范围内。这样的其他方法会比图 5 和图 6 中的简单视图复杂，但是它们执行类似的功能，实现同样的整体效果。下面简要地说明编程接口的一些演示性的替换实现。

#### A. 分解

从一个代码段到另一个代码段的通讯可以通过将通讯划分为多个离散的通讯间接地实现。这在图 7 和图 8 中示意性地展示。如所示，某些接口可以根据可划分的功能集合来描述。因此，可以分解图 5 和图 6 的接口功能来实现相同的效果，就像可以在数学上提供  $24$  或  $2 \times 2 \times 3 \times 2$ 。因此，如图 7 所示，可以划分由接口 1 提供的功能以将该接口的通讯转换到多个接口中：接口 1A、接口 1B、接口 1C 等等，而实现相同的效果。如图 8 所示，由接口 I1 提供的功能可以划分为多个接口 I1a、I1b、I1c 等等而实现相同的效果。类似地，第二个代码段的接收来自第一个代码段的信息的接口 I2 可以分解为多个接口：I2a、I2b、I2c 等等。当分解时，包括在第一个代码段内的接口数量

不需要和包括在第二个代码段中的接口数量匹配。在图 7 与图 8 中，接口 1 和 I1 的功能大意各自保持和图 5 与图 6 相同。接口的分解也可以遵循关联、交换和其他数学属性以使得分解难以识别。例如，操作的顺序可能不重要，并且因此由一个接口执行的功能可以在达到该接口之前由另外的代码或接口执行，或由系统的独立组件执行。再者，有普通编程技术水平的人可以理解，存在做出实现相同效果的不同函数调用的各种方法。

### B. 定义

在某些情况下，可能忽略、增加或修订编程接口的特定方面（如参数）而仍然实现预定结果。这在图 9 和图 10 中展示。例如，假设图 5 的接口 1 包括函数调用 *Square(input, precision, output)*，该调用包括三个参数，*input*、*precision* 和 *output*，且从第一个代码段将其提交到第二个代码段。如果在给定的情景下不关心中间的参数 *precision*，则如图 9 所示，可以忽略它或甚至用无意义的参数（在此情况）来替换它。也可以增加不关心的附加参数。在两种情况中，平方的功能都可以实现，只要在输入由第二个代码段平方后返回输出。对某些下游或计算系统的其他部分，*Precision* 可能是有意义的参数；然而，一旦认为对计算平方的有限目的不需要 *precision*，则可以替换它或忽略它。例如，可以传递无意义的值，如出生率，而不是传递合法的 *precision* 值，但不会对结果产生负面影响。类似地，如图 10 所示，接口 I1 由接口 I1' 替换，修订为忽略或增加参数到该接口。接口 I2 可以类似地修订为接口 I2'，修订为忽略不需要的参数或可以在别处处理的参数。这里的关键是在某些情况下编程接口可以包括对某些目的不需要的方面，如参数，从而对其他目的可以忽略或修订或在别处处理它们。

### C. 内联编码

合并两个独立代码模块的部分或全部功能，以使得它们之间的“接口”改变形式也是可行的。例如，图 5 和 6 的功能各自可以转换为图 11 和 12 的功能。在图 11 中，前面图 5 中的第一个和第二个代码段合并为一个包含两者的模块。在此情况，代码段仍然彼此通讯，但是接口可能采取更加适合于单个模块的形式。因此，例如，可能不再需要正式的调用和返回语句，但是按照接口 1 进行的类似处理或响应仍然有效。类似地，如图 12 所示，图 6 的接

口 I2 的部分(或全部)可以内联写入到接口 I1 中以构成接口 I1”。如所示，接口 I2 被划分为 I2a 和 I2b，接口部分 I2a 已用接口 I1 内联编码来构成接口 I1”。对实际的例子，考虑图 6 的接口 I1 执行函数调用 `square(input, output)`，它由接口 I2 接收，接口 I2 在用第二个代码段处理通过 `input` 传递的值(对其进行平方)之后，用 `output` 传回平方的结果。在这样的情况下，由第二个代码段执行的处理(平方 `input`)可以由第一个代码段执行而不调用该接口。

#### D. 分离

从一个代码段到另一个代码段的通讯可以通过将通讯分解为多个离散的通讯间接地实现。这在图 13 和图 14 中展示。如图 13 所示，提供一个或多个中间件(分离接口，由于它们从初始的接口分离功能和/或接口函数)以转换第一个接口(接口 1)上的通讯，使之符合不同的接口，在此例中为接口 2A、接口 2B 和接口 2C。在安装了设计为根据接口 1 的协议和操作系统通讯的应用，但是操作系统改变为使用不同的接口(在此例中为接口 2A、接口 2B 和接口 2C)时，可以实现这样的结构。关键是由第二个代码段使用的初始的接口发生了改变，从而它不再和第一个代码段使用的接口兼容，因此使用中间层以使得旧接口和新接口兼容。类似地，如图 14 所示，可以引入第三个代码段，从而用分离接口 DI1 接收来自接口 I1 的通讯，用分离接口 DI2 发送接口功能到，例如，重新设计为对 DI2 工作但是提供相同的功能效果的接口 I2a 和 I2b。类似地，DI1 和 DI2 可以共同工作以将图 6 的接口 I1 和 I2 的功能转换到新的操作系统，而提供相同或近似的功能效果。

#### E. 重写

另一种可能的情况是动态地重写代码以用其他实现相同总体效果的代码替换接口功能。例如，存在这样的系统，它们在(如由 .Net 框架、Java 运行库环境，或其他类似的运行时类型环境提供的)执行环境中向运行时编译执行的(JIT)编译器或解释器提供以中间语言(如 Microsoft IL、Java 字节码等等)出现的代码段。可以写出 JIT 编译器，以动态地转换从第一个代码段到第二个代码段的通讯，即，符合如第二个代码段(初始的或不同的第二个代码段)所要求的不同的接口。这在图 15 和 16 中展示。如图 15 所示，此方法类似于上述的分离情况。在安装了设计为根据接口 1 的协议和操作系统通讯的应用，

但是操作系统改变为使用不同的接口时，可以实现这样的结构。可以使用 JIT 编译器即时使得来自安装的应用的通讯符合操作系统的新的接口。如图 16 所示，这种动态地重写接口的方法可以应用于动态的因素，或相反地修改这些接口。

应注意通过替换实施例实现和接口相同或类似的效果的上述情景也可以根据各种方法进行组合，串行和/或并行、或用其他中介代码。因此，上述替换实施例不是互斥的，且可以混合、匹配并合并，以产生与图 5 和图 6 所展示的通用情景相同或等价的情景。也应注意，对多数编程结构，存在实现和接口相同或类似功能的其他类似方法，没有在此对他们进行说明，但是它们由本发明的精神和范围所代表，即，应注意它至少是由支撑接口价值的接口表示的部分功能及其带来的优点。

## 结论

虽然已用特定于结构特性和/或方法动作的语言对本发明进行了说明，应理解在所附的权利要求中定义的本发明不限于所述的具体特性或动作。相反，所揭示的具体特性和动作是实现所请求的发明的示范形式。

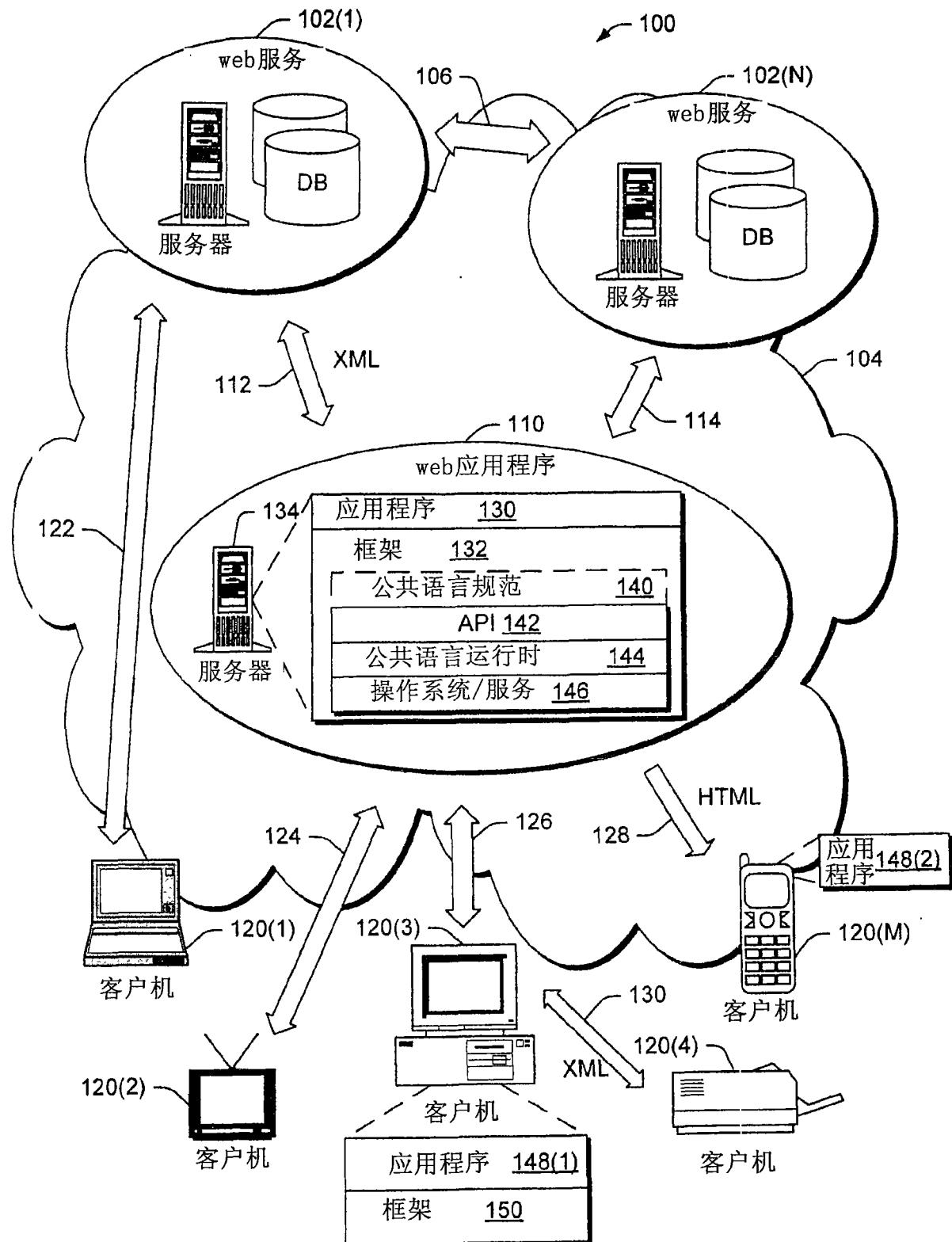


图 1

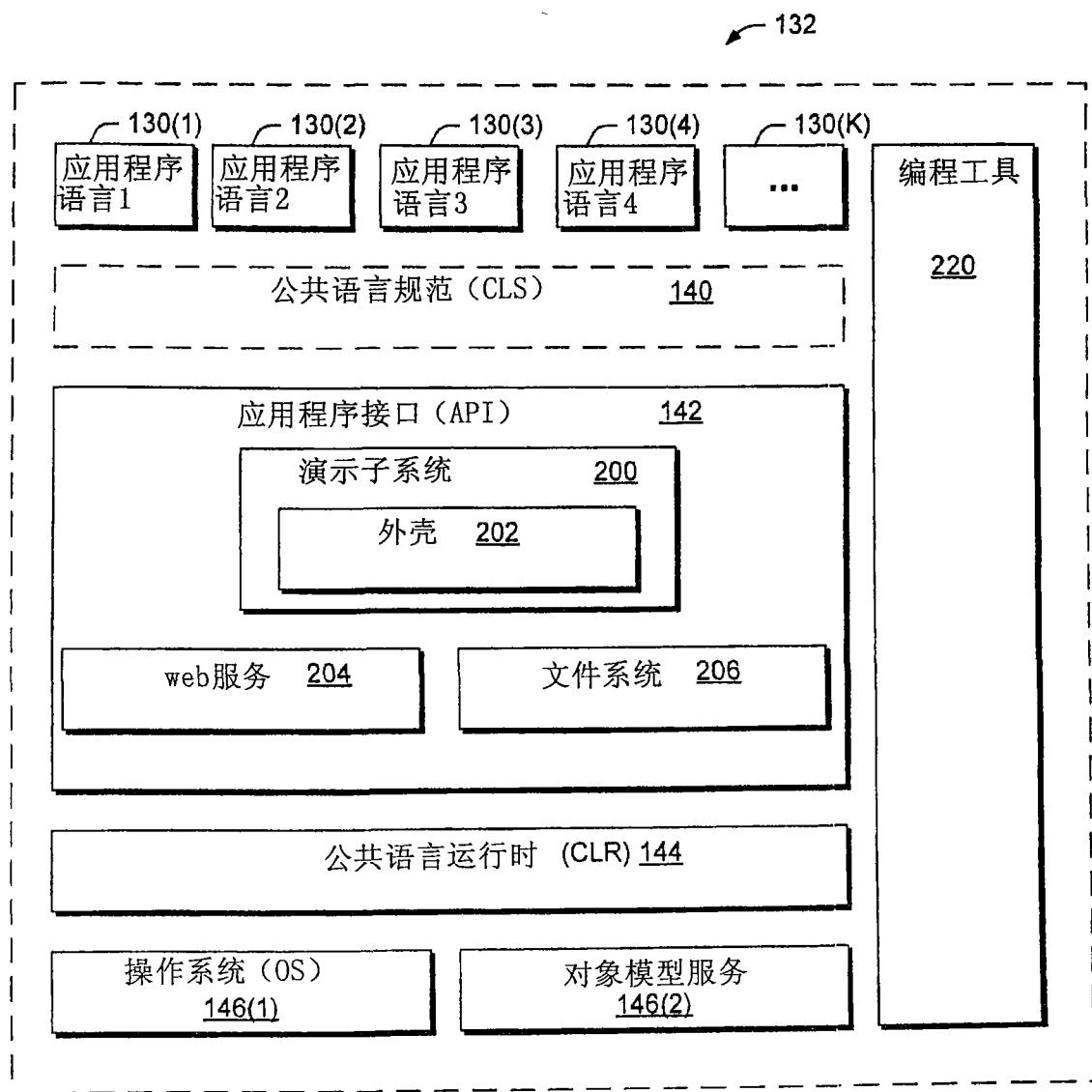


图 2

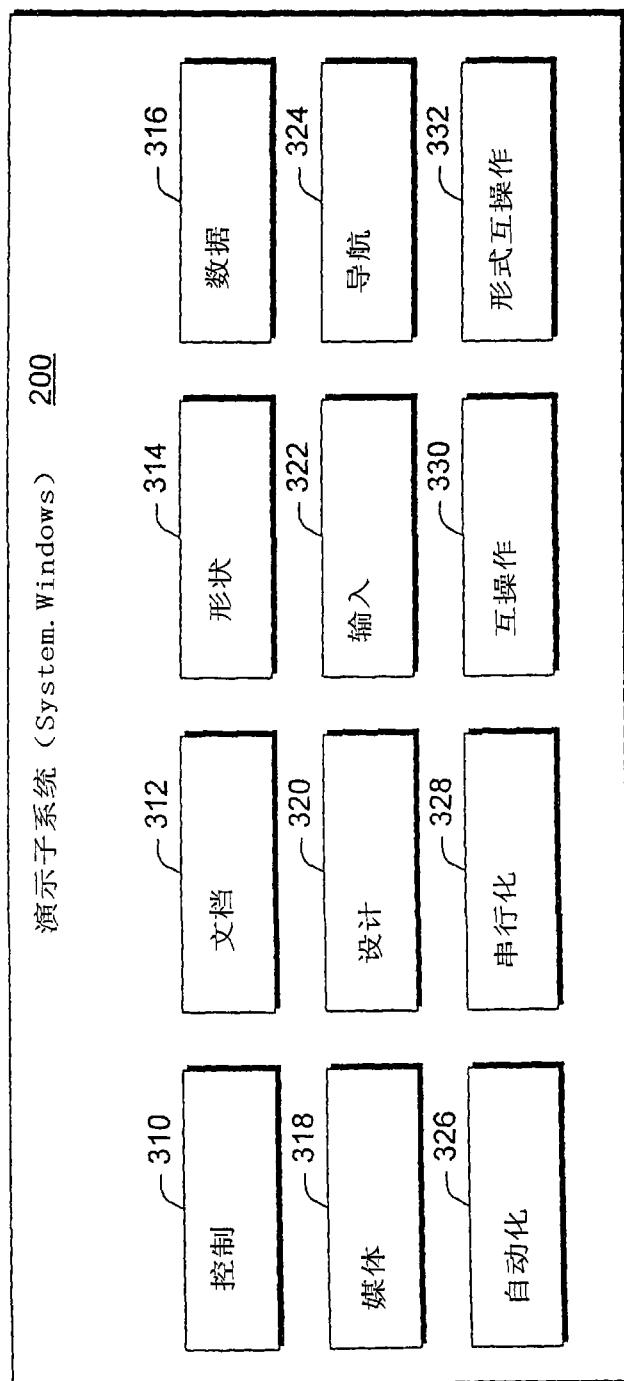


图 3

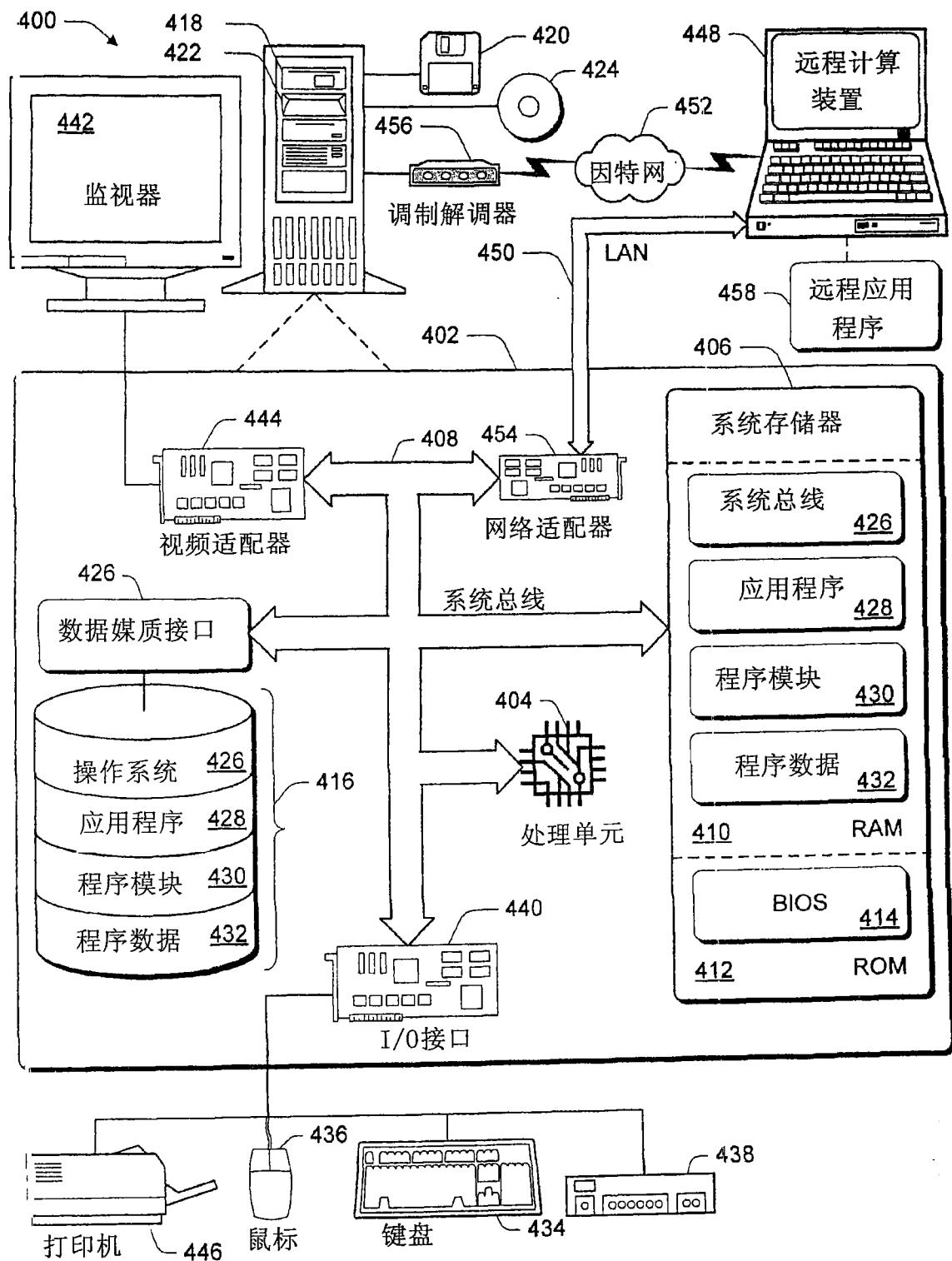


图 4

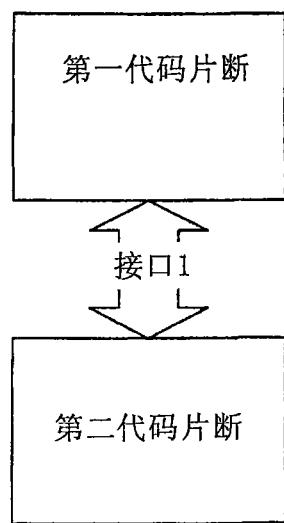


图 5

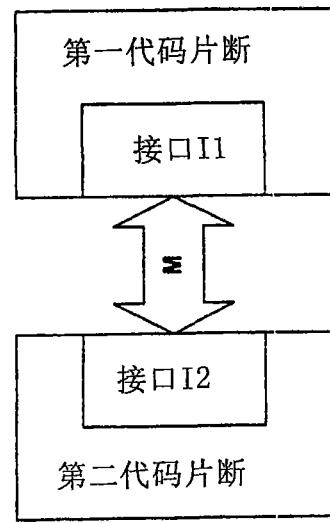


图 6

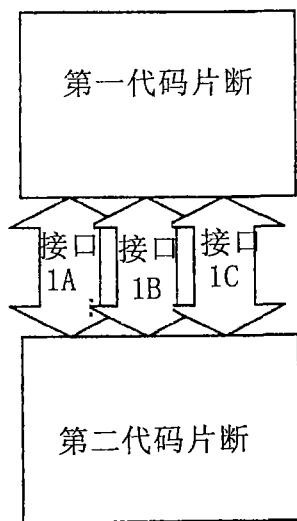


图 7

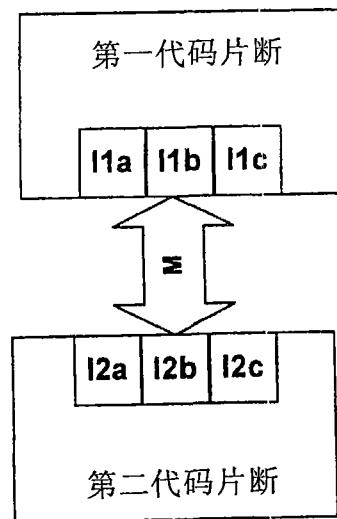


图 8

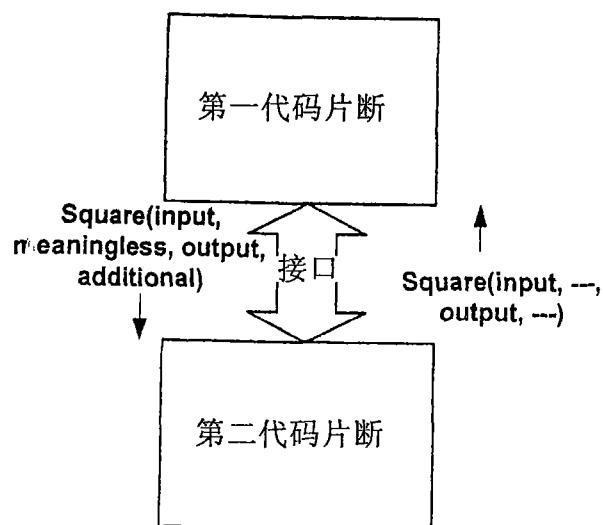


图 9

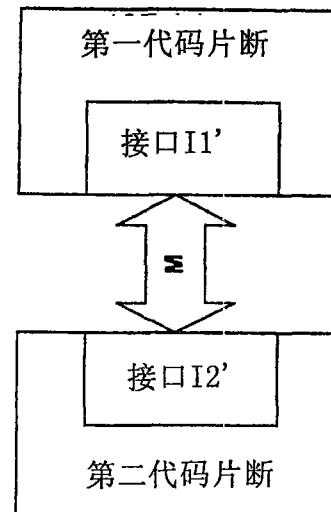


图 10

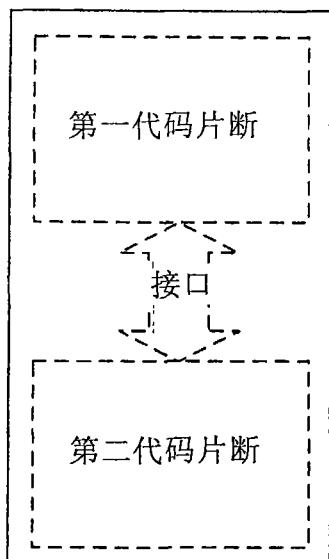


图 11

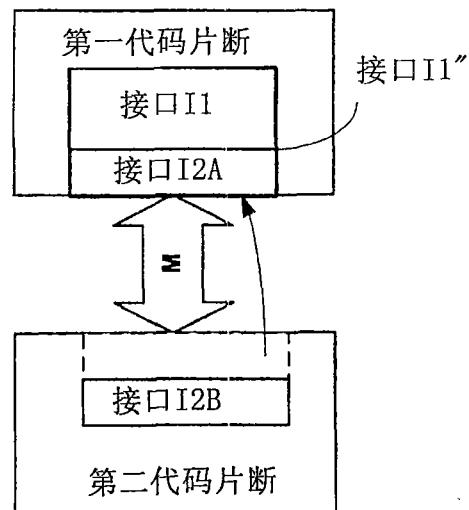


图 12

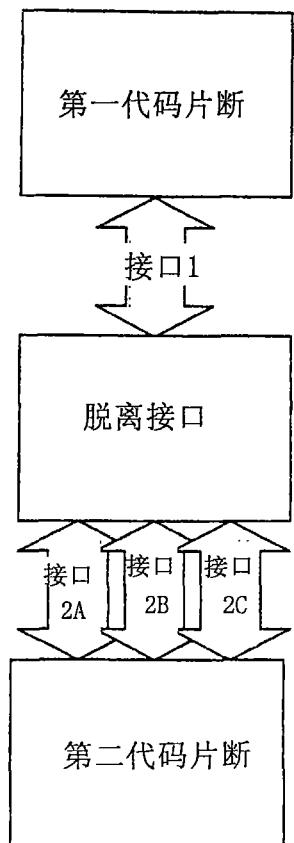


图 13

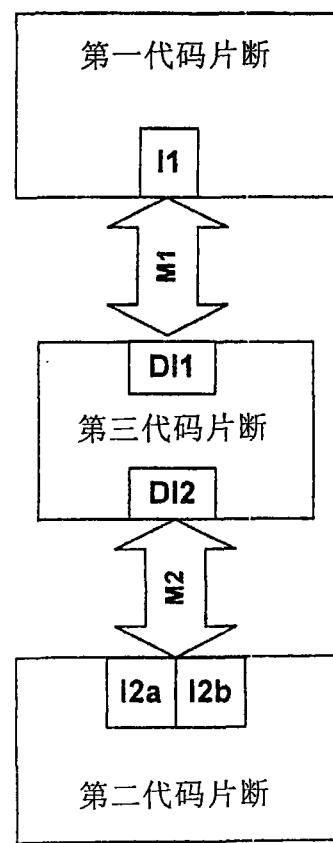
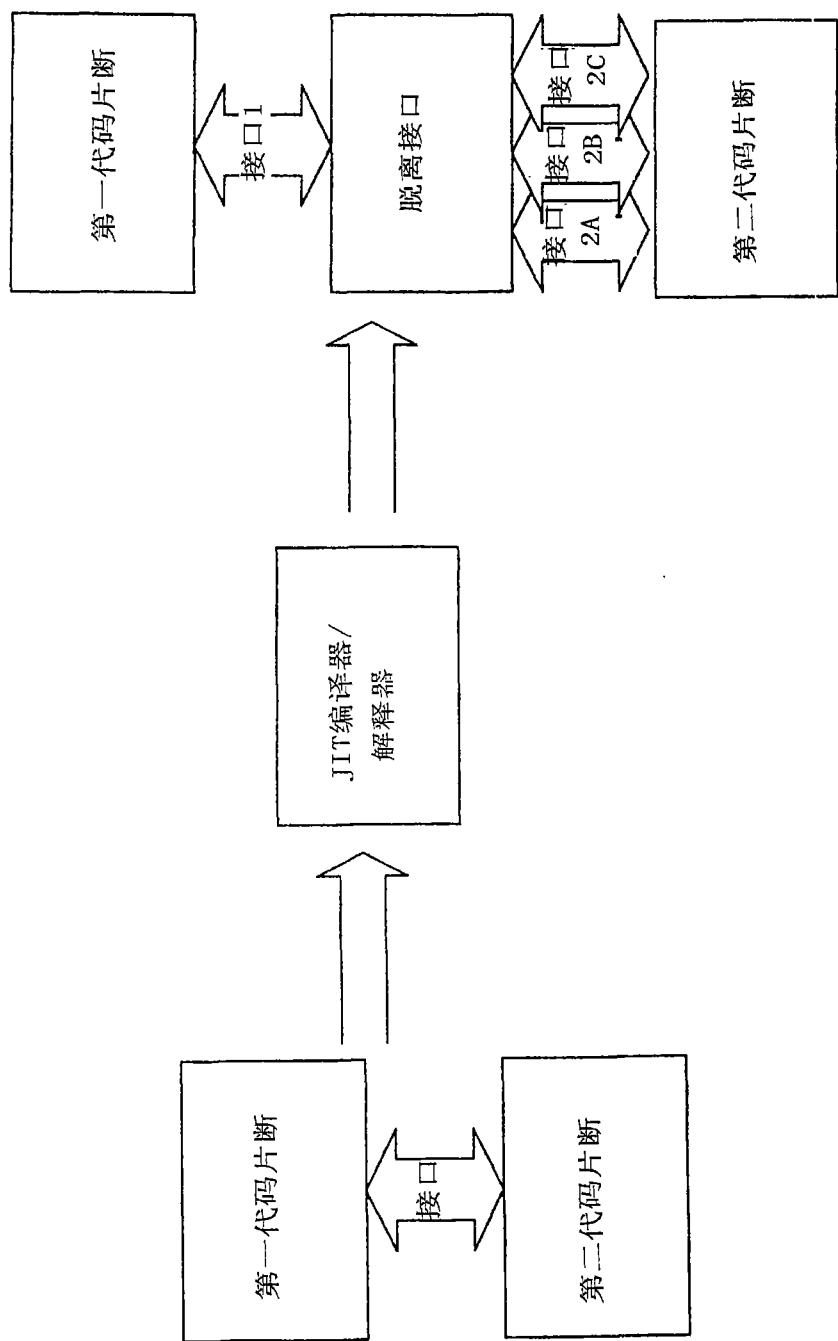


图 14



15

图

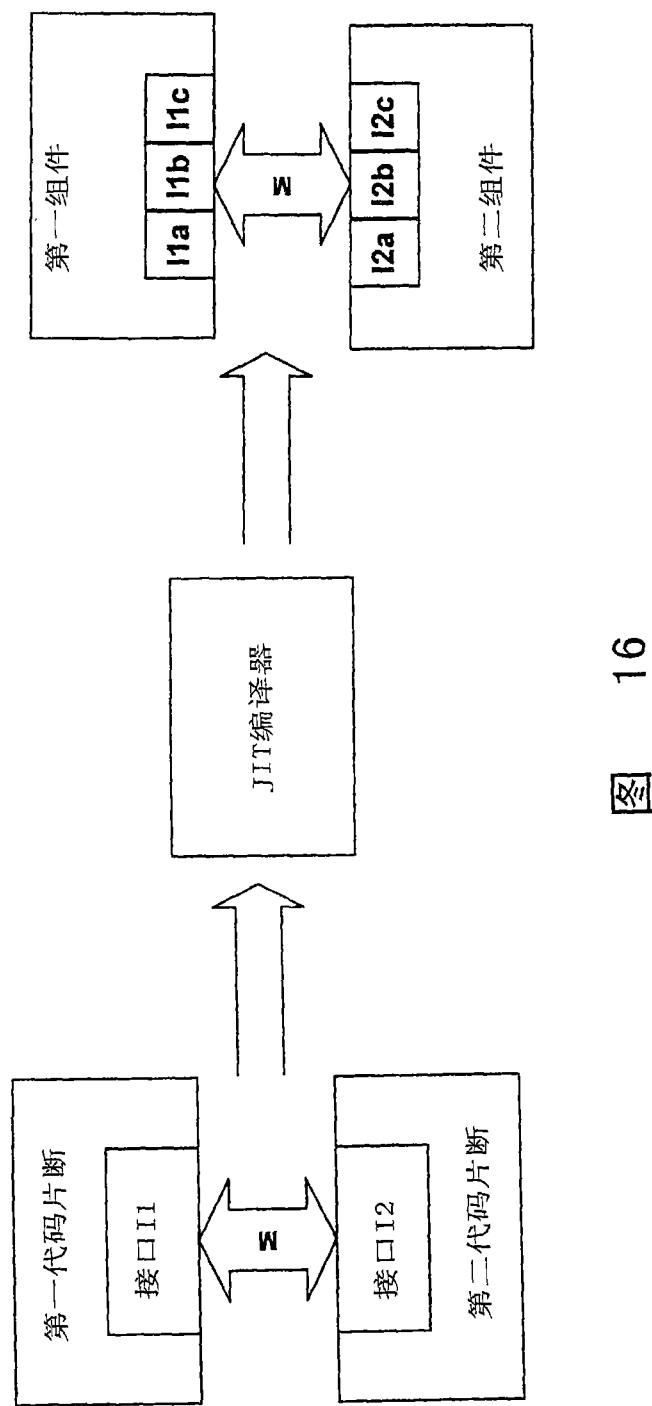


图 16