

493

93

SUPERINDUSTRIA Y COMERCIO

Radicación : 04090117 03080602

Folios: 412

Fecha (RE): 2004-12-02 16:23:11

Trámite : 002 PATENTES D 1 REGISTRO/ 444 REMUESTA

Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

Señores

**SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO
DIVISIÓN DE NUEVAS CREACIONES**

E.

S.

D.

Ref.: Expediente No. 04-090.117

Solicitud de Patente para "INTERFASE DE PROGRAMACIÓN
PARA UNA PLATAFORMA DE COMPUTO".

Solicitante: Microsoft Corporation.

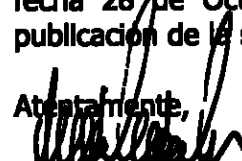
Darío Cárdenas Navas, identificado como aparece al pie de mi firma, atentamente me dirijo a ese Despacho dando respuesta al Oficio No. 10011 de fecha 28 de Octubre de 2004.

Para tal efecto me permito adjuntar los siguientes documentos:

1. Copia auténtica y debidamente Apostillada del documento donde consta la cesión del derecho a la patente otorgada por los Inventores a favor del solicitante.
2. Copia auténtica del documento de prioridad correspondiente a la patente de los Estados Unidos No. 10.694.080 de fecha 24 de Octubre de 2003. Tal como consta en los archivos de la Oficina de Patentes, el recibo oficial 04-072.022 de fecha 2 de Septiembre de 2004 correspondiente a la reivindicación de la prioridad fue presentado junto con la solicitud de patente radicada ante ese Despacho el día 10 de Septiembre de 2004.

Habiendo dado cumplimiento a los requerimientos incluidos en el Oficio No. 10011 de fecha 28 de Octubre de 2004 atentamente solicito a ese Despacho proceda con la publicación de la solicitud de patente en referencia.

Atentamente,


Darío Cárdenas Navas.

c.c. 17.066.629 de Bogotá

t.p. 1.250 C.S.J. 

424
504

APOSTILLE
(Convention de La Haye du 5 octobre 1961)

1. Country: United States of America

This public document

2. has been signed by P. Swain

3. acting in the capacity of Certifying Officer

4. bears the seal/stamp of Patent and Trademark Office

Certified

5. at Washington, D.C.

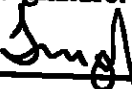
6. the fourth of October, 2004

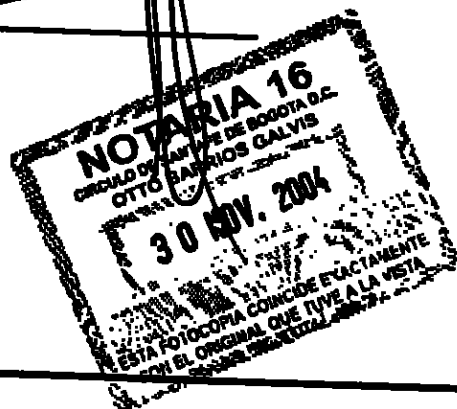
7. by Assistant Authentication Officer, United States Department of State

8. No. 05000258-1

9. Seal/Stamp:

10. Signature:


Sonya N. Johnson



485
505



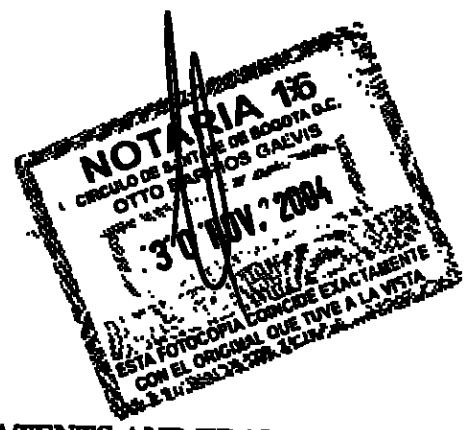
THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

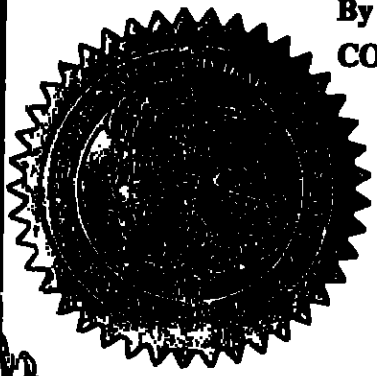
UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

September 13, 2004

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE
RECORDS OF THIS OFFICE OF A DOCUMENT RECORDED ON
May 24, 2004.



By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS



P. SWAIN
Certifying Officer

13

506

Form PTO-1055
(Rev. 03/01)
OMB No. 0751-0027 (exp. 3/31/2002)
Tab #1055 1055

**RECORDATION FORM COVER SHEET
PATENTS ONLY**

U.S. DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office

To the Honorable Commissioner of Patents and Trademarks: Please record the attached original documents or copy thereof.

1. Name of conveying party(ies): Rob Relyea Jeff Bogdan Additional name(s) of conveying party(ies) attached? ☐ Yes ☒ No	2. Name and address of receiving party(ies) Name: <u>Microsoft Corporation</u> Internal Address: _____ Street Address: <u>One Microsoft Way</u> City: <u>Redmond</u> State: <u>WA</u> Zip: <u>98052-6399</u> Additional name(s) & address(es) attached? ☐ Yes ☒ No
3. Nature of conveyance: ☒ Assignment ☐ Merger ☐ Security Agreement ☐ Change of Name ☐ Other: _____ Execution Date: <u>02/4/04</u>	4. Application number(s) or patent number(s): <u>10/694,080</u> If this document is being filed together with a new application, the execution date of the application is: _____ A. Patent Application No.(s) _____ B. Patent No.(s) _____ Additional numbers attached? ☐ Yes ☒ No
5. Name and address of party to whom correspondence concerning document should be mailed: Name: <u>Steven R. Sponseller, Reg. No. 38,384</u> Internal Address: <u>Lee & Hayes, PLLC</u> Street Address: <u>421 West Riverside Avenue</u> <u>Suite 500</u> City: <u>Spokane</u> State: <u>WA</u> Zip: <u>99201</u>	6. Total number of applications and patents involved: <u>1</u> 7. Total fee (37 CFR 3.41):\$ <u>40.00</u> ☐ Enclosed ☒ Authorized to be charged to deposit account 8. Deposit account number: <u>12-0789</u> (Attach duplicate copy of this page if paying by deposit account)
8. Statement and signature. To the best of my knowledge and belief, the foregoing information is true and correct, and the attached copy is a true copy of the original document. <u>Steven R. Sponseller, Reg. No. 38,384</u> Name of Person Signing Signature: <u>[Signature]</u> Total number of pages including cover sheet, attachments and documents: <u>3</u> Mail documents to be recorded with required cover sheet about 10 business days before recording. Mail Stop Assignment Recording Services Alexandria, VA 22313-1402	

DO NOT USE THIS SPACE

NOTARY PUBLIC
CIRCULO DE SELLADO DE BOBROS
OTTO B. ALVAREZ
38 NOV 2004
PATENT
REEL: 014665 FRAME: 0388

700086844

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number 10/694,080
Filing Date Oct 24, 2003
Inventorship Relyea et al.
Applicant Microsoft Corporation
Attorney's Docket No. MS1-1779US
MSDocket No. 306459.01
Title: Programming Interface for a Computer Platform

PATENT ASSIGNMENT

PARTIES TO THE ASSIGNMENT

Assignor(s):

Rob Relyea
4618 142nd Pl SE
Bellevue, WA 98006

Jeff Bogdan
3042 125th Ave. NE
Bellevue, WA 98005

Assignee:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

AGREEMENT

WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention entitled "Programming Interface for a Computer Platform," as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft Corporation (hereinafter referred to as "Assignee"), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

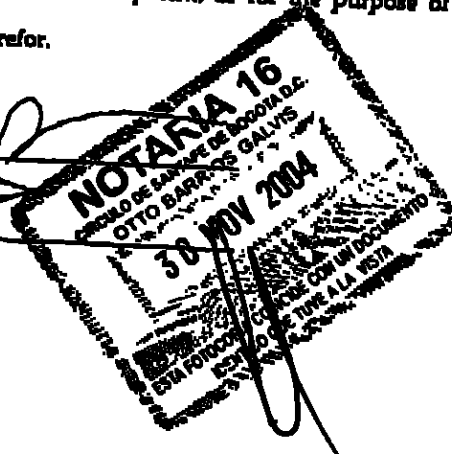
NOTARIAL PUBLIC
STATE OF WASHINGTON
COMMISSION EXPIRES 12/31/04
NOTARY PUBLIC
STATE OF WASHINGTON
COMMISSION EXPIRES 12/31/04

Attorney Docket No. MS1-1779US
MS Docket No. 306439.01
Serial No. (if known) 10/184,080

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made. ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue all letters patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all instruments and documents required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent therefor.

5/24/2007
Date
5/24/2004
Date

Rob Relyea
Jeff Bogdan



RECORDED: 05/24/2004

Page 2 of 2

PATENT
REEL: 014665, FRAME: 0390

490
509

TRADUCCIÓN OFICIAL

Traducción oficial de un documento en inglés. Para su identificación se sella con el sello de la traductora. Esta traducción ha sido preparada por Miriam Cotes, traductora oficial debidamente reconocida por el Ministerio de Relaciones Exteriores y autorizada por el Ministerio de Justicia, Res. Número 3779 de noviembre de 1993.

APOSTILLE

(Convención de La Haya del 5 de octubre de 1961)

1. País: Estados Unidos de América
2. Este documento público ha sido firmado por: P. Swain
3. Actuando como: Funcionario de Certificaciones
4. Con el sello/estampilla de: Oficina de Patentes y Marcas

Certificado

5. en Washington D.C.
6. el cuatro de octubre de 2004
7. por el Funcionario Asistente de Certificaciones
8. No. 05000258-1
9. Sello/estampilla:
10. Firma: Sonya N. Johnson

THIS IS A COMPLETE AND EXACT TRANSLATION. FOR ITS CONFRONTATION A COPY OF IT IS KEPT IN THIS OFFICE.

Bogotá, November 22, 2004

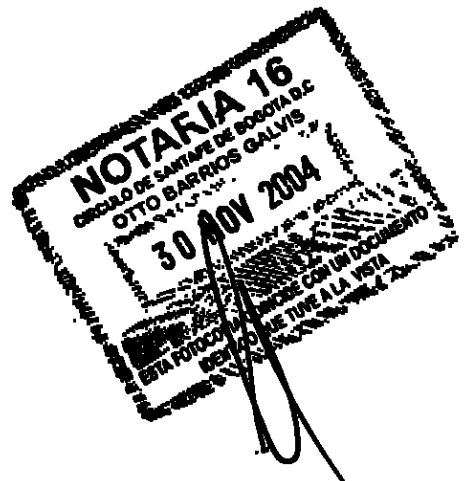
Number of pages including this one: 1

NO SE ASUME
RESPONSABILIDAD DEL TEXTO

2004 NOV 29 P 1:55

les

JEFE DE LEGALIZACIONES
SANTAFE DE ESPEDA D.C.



MINISTERIO DE RELACIONES
EXTERIORES

bb 248
Miriam Cotes
CUYA FIRMA APARECE EN EL
DOCUMENTO DESEMPENA
LAS FUNCIONES DE TRADUCCION

Miriam Cotes Benitez

516

TRADUCCIÓN OFICIAL

Traducción oficial de un documento en inglés. Para su identificación se sella con el sello de la traductora. Esta traducción ha sido preparada por Miriam Cotes, traductora oficial debidamente reconocida por el Ministerio de Relaciones Exteriores y autorizada por el Ministerio de Justicia, Res. Número 3779 de noviembre de 1993.

LOS ESTADOS UNIDOS DE AMÉRICA
A todos aquellos a los que se presente este documento:

EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS
La Oficina de Patentes y Marcas de los Estados Unidos

Septiembre 13, 2004

ESTE DOCUMENTO CERTIFICA QUE EL ANEXO A LA PRESENTE ES UNA COPIA AUTÉNTICA DEL DOCUMENTO REGISTRADO EN ESTA OFICINA EN MAYO 24 DE 2004.

(SELLO) Por la autoridad del
Comisionado de Patentes y Marcas

Firmado: P. Swain
Oficial de Certificaciones

THIS IS A COMPLETE AND EXACT TRANSLATION. FOR ITS CONFRONTATION A COPY OF IT IS KEPT IN THIS OFFICE.

Bogotá, November 22, 2004

Number of pages including this one: 1



Miriam Cotes Benítez

57 J.
611

TRADUCCIÓN OFICIAL

Traducción oficial de un documento en inglés. Para su identificación se sella con el sello de la traductora. Esta traducción ha sido preparada por Miriam Cates, traductora oficial debidamente reconocida por el Ministerio de Relaciones Exteriores y autorizada por el Ministerio de Justicia, Res. Número 3779 de noviembre de 1993.

Formato PTO-1595 Carátula formato de archivo DÉPARTAMENTO DE COMERCIO DE EUU (Rev. 03/01) SÓLO PATENTES	
OMB No. 0651-0027 (5/31/2002)	
Al Honorable Comisionado de Patentes y Marcas, favor registrar los documentos originales anexas al documento o sus copias.	
1. Nombre de los emisores: Rob Relyea Jeff Bogdan Otros nombres de emisores: ☐ SI ☒ No	2. Nombre de los receptores: Nombre: <u>Microsoft Corporation</u> Dirección interna: _____ Dirección postal: <u>One Microsoft Way</u> Ciudad: <u>Redmond</u> Estado: <u>WA</u> Zip: <u>98052-6399</u> ¿Nombres(s) y dirección(es) anexas? ☐ SI ☒ No
3. Naturaleza de la emisión: ☒ Cesión: ☐ Fusión: Acuerdo de Seguridad Cambio de nombre Otros _____	Fecha de ejecución: <u>5/24/2004</u>
4. Número (s) de la solicitud o números de la patente: <u>10/694.080</u> Si este documento se está archivando con una nueva solicitud, el día de ejecución de la solicitud es: A. Número (s) de la solicitud de patente B. Número (s) de la patente ¿Números adicionales anexas? ☐ SI ☒ No	
5. Nombre y dirección de la parte a quien se le debe enviar por correo la correspondencia relativa a la patente: Nombre: <u>Steven R. Sponseller, Reg. No. 39,384</u> Dirección interna: <u>Lee & Hayes, PLLC</u> Dirección Postal: <u>421 West Riverside Avenue Suite 500</u> Ciudad: <u>Spokane</u> Estado: <u>WA</u> Zip: <u>99201</u>	6. Número total de solicitudes y patentes: <u>1</u> 7. Costo total (37CFR 3.41)..... <u>\$40.00</u> ☐ Adjunto ☒ Autorización para deducir de _____ 8. Cuenta número: <u>12-0769</u> (Adjuntar copia de esta página a la cuenta de la cuenta EXACTAMENTE la cuenta QUE LLEVE A LA VISTA
NO USE ESTE ESPACIO	
9. Declaración y firma. <i>De acuerdo con mi leal saber y entender, la información aquí consignada es verdadera y correcta, y toda copia adjunta es una copia fiel del documento original.</i> <u>Steven R. Sponseller, No. Reg. 39,384</u> Firma: (ilegible) Fecha: <u>Julio 24 de 2004</u> Nombre del firmante Número total de páginas incluyendo la carátula, los anexos y los documentos: <u>3</u>	

NOTARIA 16
CAROL A. SANTANA DE BOGOTA D.C.
OTRO BARRIOS GALVIS
10 NOV. 2004

Miriam Cates Bentez

Envíe por correo los documentos para archivar con la información requerida de la carátula a:
Mail Stop Assignment Recordation Services
Alexandria, VA 22313-1450

PATENTE
CARRETE: 014665 MARCO: 0388

70086844

***THIS IS A COMPLETE AND EXACT TRANSLATION. FOR ITS
CONFRONTATION A COPY OF IT IS KEPT IN THIS OFFICE.***

Bogotá, November 5, 2004

Number of pages including this one: 2



H. Juan Cotes Benítez

513

TRADUCCIÓN OFICIAL

Traducción oficial de un documento en inglés. Para su identificación se sella con el sello de la traductora. Esta traducción ha sido preparada por Mirlam Cotes, traductora oficial debidamente reconocida por el Ministerio de Relaciones Exteriores y autorizada por el Ministerio de Justicia, Res. Número 3779 de noviembre de 1993.

Minuta del abogado No. MS1-1779US

Minuta de MS No: 306459.01

Número Serial: 10/694.080

EN LA OFICINA DE PATENTES Y MARCAS DE EEUU

Número Serial: 10/694.080

Fecha de archivo: octubre 24 de 2003

Invencción: Relyea et. al.

Solicitante: Microsoft Corporation

Minuta del abogado No. MS1-1779US

Minuta de MS: No: 306459.01

Nombre: Interfaz de programación para plataforma de computador

Cesión de patente

Cedente:

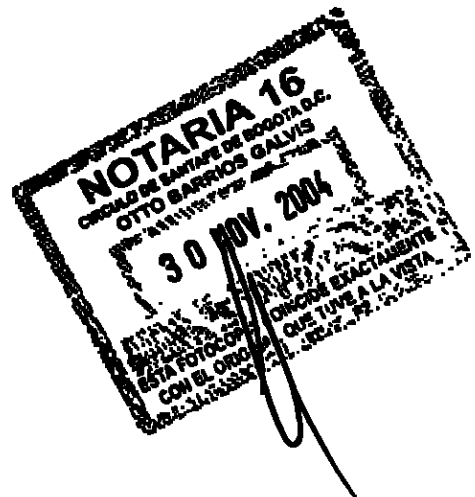
Rob Relyea
4618 142nd Pl. SE
Bellevue, WA 98006

Jeff Bodgan
3042 125th Ave. NE
Bellevue, WA 98006

Cesionario:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

ACUERDO



Mirlam Cotes Buitrago

594
514

CONSIDERANDO QUE EL/LOS CEDENTE(S) arriba mencionados son inventores de un invento llamado "Interfaz de programación para plataforma de computador" como se describe y reclama en la especificación que hace parte de una solicitud de patente en los Estados Unidos que se referencia más arriba;

CONSIDERANDO que Microsoft Corporation (de aquí en adelante llamada el CESIONARIO), una corporación del Estado de Washington que tiene una sede de negocios en One Microsoft Way, Redmond, WA 98052 desea adquirir el pleno derecho, el nombre y el interés, en el invento y en toda patente que sea concedida en los Estados Unidos o en cualquier otro país;

Los CEDENTES, a cambio de una buena y valiosa consideración, cuyo recibo se reconoce aquí, venden, ceden y transfieren al CESIONARIO el pleno derecho, nombre e interés del mencionado invento, la mencionada solicitud y todas y cada una de las patentes que se le otorguen al mencionado invento en los Estados Unidos de América y sus posesiones territoriales, y en todos los otros países, y a todas las divisiones, redicciones y continuaciones a las que haya lugar. Esto incluye el derecho de aceptar solicitudes extranjeras por parte del CESIONARIO y reclamar derechos de prioridad derivados de las solicitudes de otros países en virtud de convenios, tratados internacionales u otros. Todas las solicitudes y patentes de dicho invento serán poseídas y disfrutadas por el CESIONARIO y sus sucesores para su disfrute y beneficio tal y como la hubieran beneficiado y disfrutado los CEDENTES si esta cesión y venta no se hubiera llevado a cabo. Por medio de la presente, los CEDENTES autorizan y solicitan al Comisionado de Patentes y Marcas que expida toda patente de dicha invención al CESIONARIO. Los CEDENTES acuerdan ejecutar todos los instrumentos y documentos requeridos para la elaboración y procesamiento de solicitudes de patentes de dicho invento en los Estados Unidos y otros países, para los litigios que tengan que ver con dicha patente, o para proteger el título del mencionado invento o las patentes que se expidan en adelante.

Fecha: julio 24 de 2004

Firmado por: Rob Relyea

Fecha: julio 24 de 2004

Firmado por: Jeff Bodgan

Archivado: julio 24 de 2004

PATENTE

CARRETE: 014893 MARCO: 0491

30 NOV 2004
H. Juan Cotes Buitrago

THIS IS A COMPLETE AND EXACT TRANSLATION. FOR ITS
CONFRONTATION A COPY OF IT IS KEPT IN THIS OFFICE.

Bogotá, noviembre 24 de 2004

Number of pages including this one: 3



525
515
Mariano Coto Barrantes

516

(b)

PA 1222438



THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

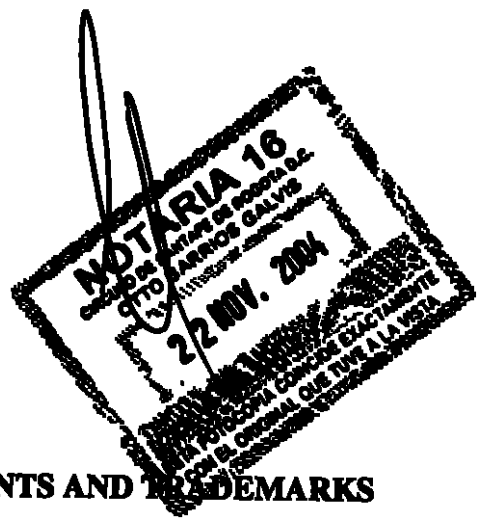
United States Patent and Trademark Office

September 16, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: 10/694,080

FILING DATE: October 24, 2003



By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS




P. SWAIN
Certifying Officer

227
517

EV355227078

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship..... Relyea et al.
Applicant..... Microsoft Corporation
Attorney's Docket No. MSI-1779US
Title: Programming Interface for a Computer Platform

22141 U.S. PTO
10/694080
102403

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

From: Steven R. Sponseller (Tel. 509-324-9256; Fax 509-323-8979)
Customer No. 22801

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification—title page, plus 188 pages, including 58 claims and Abstract
2. Transmittal letter including Certificate of Express Mailing
3. 9 Sheets Formal Drawings (Figs. 1-16)
4. Return Post Card
5. Petition to Suspend Rules Pursuant to Rule 1.183
6. Two duplicative sets of three compact discs (CD-Rs) each labeled "Copy 1" and "Copy 2".

The two sets of compact discs are identical. The compact discs, and files stored thereon, may be read using a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98, Windows® 2000, Windows® XP, etc.). The first compact disc in each set of three compact discs (CD 1 of 3) includes a file folder named "lheadk" that was created on October 22, 2003; it is 586 Mbytes in size, contains 9,692 sub-folders, and contains 44,292 sub-files. The second compact disc in each set of three compact discs (CD 2 of 3) includes a file folder named "ns" that was created on October 22, 2003; it is 605 Mbytes in size, contains 12,628 sub-folders, and contains 44,934 sub-files. The third compact disc in each set of three compact discs (CD 3 of 3) includes a file folder named "ns" that was created on October 22, 2003; it is 575 Mbytes in size, contains 9,881 sub-folders, and contains 43,630 sub-files.

Date: 10/24/2003

By: Steven R. Sponseller
Reg. No. 39,384

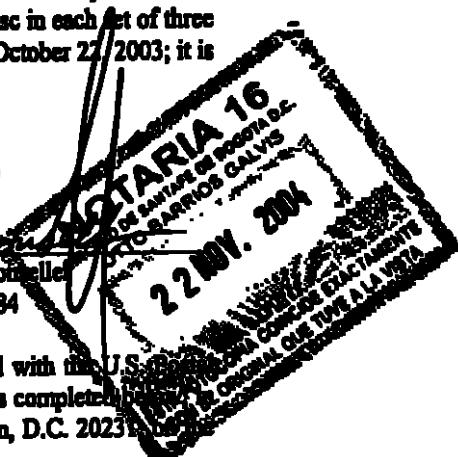
CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EV355227078

Date: 10/24/03

By: Carlin Taylor
Carlin Taylor



15

518

EV355227078

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

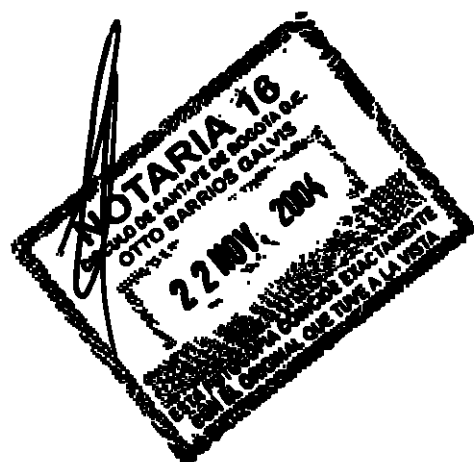
APPLICATION FOR LETTERS PATENT

Programming Interface for a Computer Platform

Inventor(s):

Rob Relyea

Jeff Bogdan



ATTORNEY'S DOCKET NO. MS1-1779US

16

519

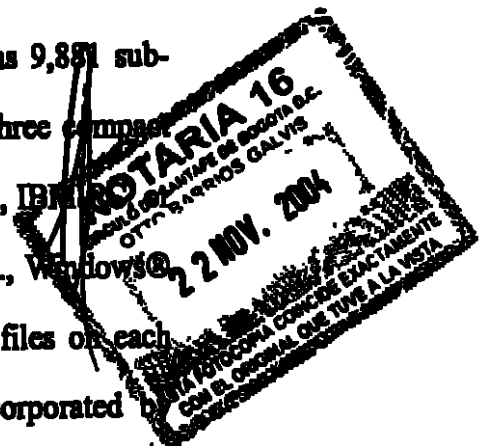
TECHNICAL FIELD

This invention relates to software and to development of such software. More particularly, this invention relates to a programming interface that facilitates use of a software platform by application programs and computer hardware.

BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISCS

Accompanying this specification is a set of three compact discs that stores a Software Development Kit (SDK) for the Microsoft® Windows® Code-Named "Longhorn" operating system. The SDK contains documentation for the Microsoft® Windows® Code-Named "Longhorn" operating system. Duplicate copies of each of these three compact discs also accompany this specification.

The first compact disc in the set of three compact discs (CD 1 of 3) includes a file folder named "lhasdk" that was created on October 22, 2003; it is 586 Mbytes in size, contains 9,692 sub-folders, and contains 44,292 sub-files. The second compact disc in the set of three compact discs (CD 2 of 3) includes a file folder named "ns" that was created on October 22, 2003; it is 605 Mbytes in size, contains 12,628 sub-folders, and contains 44,934 sub-files. The third compact disc in the set of three compact discs (CD 3 of 3) includes a file folder named "ns" that was created on October 22, 2003; it is 575 Mbytes in size, contains 9,881 sub-folders, and contains 43,630 sub-files. The files on each of these three compact discs can be executed on a Windows®-based computing device (e.g., IBM® equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98, Windows® 2000, Windows® XP, etc.). The files on each compact disc in this set of three compact discs are hereby incorporated by reference.



12

510
520

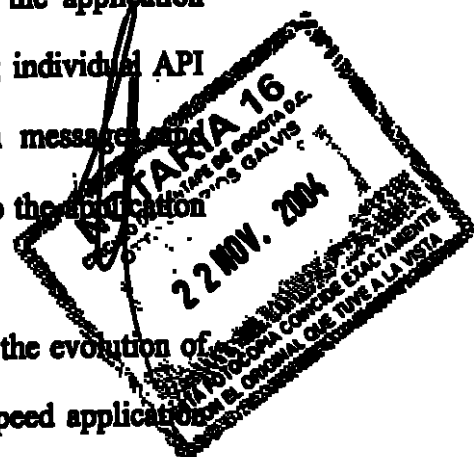
Each compact disc in the set of three compact discs itself is a CD-R, and conforms to the ISO 9660 standard. The contents of each compact disc in the set of three compact discs is in compliance with the American Standard Code for Information Interchange (ASCII).

BACKGROUND

Very early on, computer software came to be categorized as "operating system" software or "application" software. Broadly speaking, an application is software meant to perform a specific task for the computer user such as solving a mathematical equation or supporting word processing. The operating system is the software that manages and controls the computer hardware. The goal of the operating system is to make the computer resources available to the application programmer while at the same time, hiding the complexity necessary to actually control the hardware.

The operating system makes the resources available via functions that are collectively known as the Application Program Interface or API. The term API is also used in reference to a single one of these functions. The functions are often grouped in terms of what resource or service they provide to the application programmer. Application software requests resources by calling individual API functions. API functions also serve as the means by which messages and information provided by the operating system are relayed back to the application software.

In addition to changes in hardware, another factor driving the evolution of operating system software has been the desire to simplify and speed application software development. Application software development can be a daunting task,



18

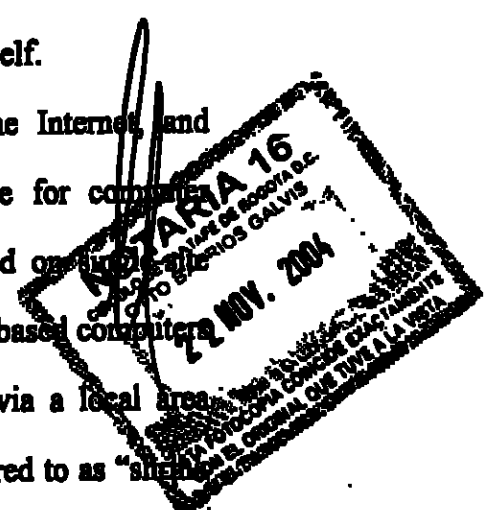
521

sometimes requiring years of developer time to create a sophisticated program with millions of lines of code. For a popular operating system such as various versions of the Microsoft Windows® operating system, application software developers write thousands of different applications each year that utilize the operating system. A coherent and usable operating system base is required to support so many diverse application developers.

Often, development of application software can be made simpler by making the operating system more complex. That is, if a function may be useful to several different application programs, it may be better to write it once for inclusion in the operating system, than requiring dozens of software developers to write it dozens of times for inclusion in dozens of different applications. In this manner, if the operating system supports a wide range of common functionality required by a number of applications, significant savings in applications software development costs and time can be achieved.

Regardless of where the line between operating system and application software is drawn, it is clear that for a useful operating system, the API between the operating system and the computer hardware and application software is as important as efficient internal operation of the operating system itself.

Over the past few years, the universal adoption of the Internet, and networking technology in general, has changed the landscape for computer software developers. Traditionally, software developers focused on writing software applications for standalone desktop computers, or LAN-based computers that were connected to a limited number of other computers via a local area network (LAN). Such software applications were typically referred to as "shrink-wrapped" products because the software was marketed and sold in a shrink-



17

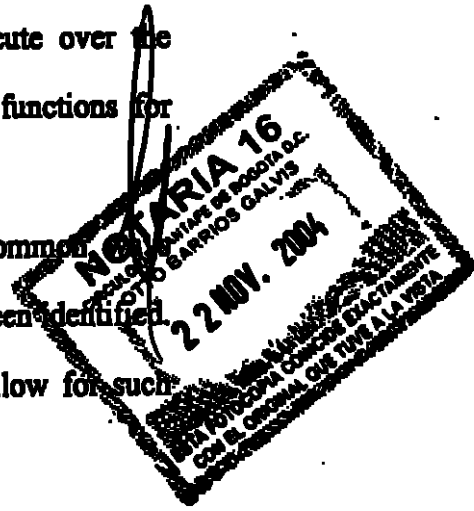
522

1 wrapped package. The applications utilized well-defined APIs to access the
2 underlying operating system of the computer.

3 As the Internet evolved and gained widespread acceptance, the industry
4 began to recognize the power of hosting applications at various sites on the World
5 Wide Web (or simply the "Web"). In the networked world, clients from anywhere
6 could submit requests to server-based applications hosted at diverse locations and
7 receive responses back in fractions of a second. These Web applications, however,
8 were typically developed using the same operating system platform that was
9 originally developed for standalone computing machines or locally networked
10 computers. Unfortunately, in some instances, these applications do not adequately
11 transfer to the distributed computing regime. The underlying platform was simply
12 not constructed with the idea of supporting limitless numbers of interconnected
13 computers.

14 To accommodate the shift to the distributed computing environment being
15 ushered in by the Internet, Microsoft Corporation developed a network software
16 platform known as the ".NET" Framework (read as "Dot Net"). Microsoft® .NET
17 is software for connecting people, information, systems, and devices. The
18 platform allows developers to create Web services that will execute over the
19 Internet. This dynamic shift was accompanied by a set of API functions for
20 Microsoft's .NET™ Framework.

21 As use of the .NET™ Framework has become increasingly common
22 to increase the efficiency and/or performance of the platform have been identified.
23 The inventors have developed a unique set of API functions to allow for such
24 increased efficiency and/or performance.



20

523

SUMMARY

A programming interface provides functions for generating applications, documents, media presentations and other content. These functions allow developers to obtain services from an operating system, object model service, or other system or service. In one embodiment, the functions allow a developer to generate a graphical user interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for a network platform, which includes an application program interface (API).

Fig. 3 is a block diagram of the presentation subsystem supported by the API, as well as function classes of the various API functions.

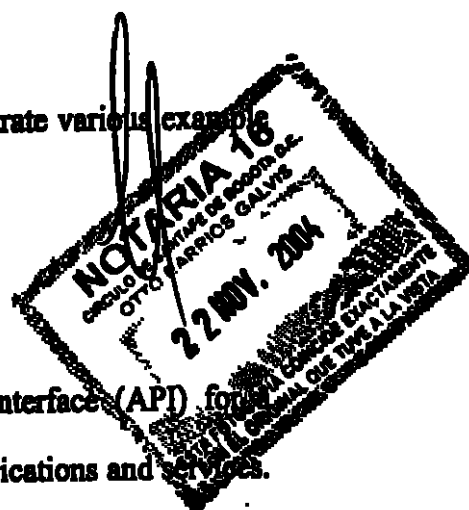
Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

Figs. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16 illustrate various example implementations of a programming interface.

DETAILED DESCRIPTION

This disclosure addresses an application program interface (API) for a network platform upon which developers can build Web applications and services.

More particularly, an exemplary API is described for operating systems that make



21

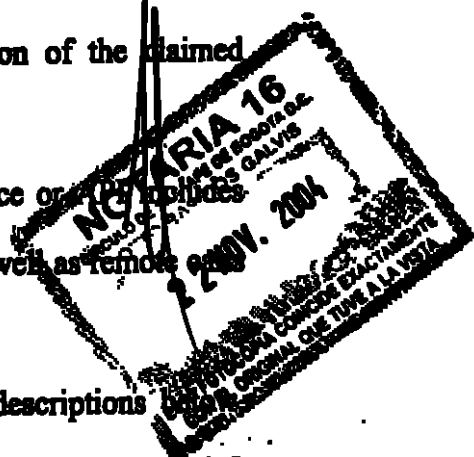
524

1 use of a network platform, such as the .NET™ Framework created by Microsoft
2 Corporation. The .NET™ Framework is a software platform for Web services and
3 Web applications implemented in the distributed computing environment. It
4 represents the next generation of Internet computing, using open communication
5 standards to communicate among loosely coupled Web services that are
6 collaborating to perform a particular task.

7 In the described implementation, the network platform utilizes XML
8 (extensible markup language), an open standard for describing data. XML is
9 managed by the World Wide Web Consortium (W3C). XML is used for defining
10 data elements on a Web page and business-to-business documents. XML uses a
11 similar tag structure as HTML; however, whereas HTML defines how elements
12 are displayed, XML defines what those elements contain. HTML uses predefined
13 tags, but XML allows tags to be defined by the developer of the page. Thus,
14 virtually any data items can be identified, allowing Web pages to function like
15 database records. Through the use of XML and other open protocols, such as
16 Simple Object Access Protocol (SOAP), the network platform allows integration
17 of a wide range of services that can be tailored to the needs of the user. Although
18 the embodiments described herein are described in conjunction with XML and
19 other open standards, such are not required for the operation of the claimed
20 invention.

21 As used herein, the phrase application program interface or API includes
22 traditional interfaces that employ method or function calls, as well as remote
23 (e.g., a proxy, stub relationship) and SOAP/XML invocations.

24 It should be appreciated that in some of namespace descriptions
25 descriptions of certain classes, interfaces, enumerations and delegates are left



22

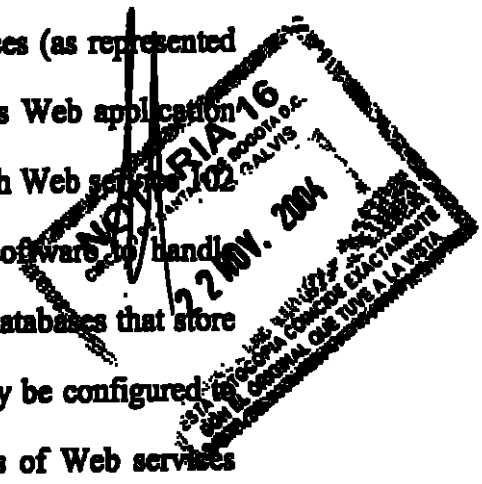
525

1 blank. More complete descriptions of these classes, interfaces, enumerations and
2 delegates can be found in the subject matter of the compact discs that store the
3 SDK referenced above.

4 5 EXEMPLARY NETWORK ENVIRONMENT

6 Fig. 1 shows a network environment 100 in which a network platform, such
7 as the .NET™ Framework, may be implemented. The network environment 100
8 includes representative Web services 102(1), ..., 102(N), which provide services
9 that can be accessed over a network 104 (e.g., Internet). The Web services,
10 referenced generally as number 102, are programmable application components
11 that are reusable and interact programmatically over the network 104, typically
12 through industry standard Web protocols, such as XML, SOAP, WAP (wireless
13 application protocol), HTTP (hypertext transport protocol), and SMTP (simple
14 mail transfer protocol) although other means of interacting with the Web services
15 over the network may also be used, such as Remote Procedure Call (RPC) or
16 object broker type technology. A Web service can be self-describing and is often
17 defined in terms of formats and ordering of messages.

18 Web services 102 are accessible directly by other services (as represented
19 by communication link 106) or a software application, such as Web application
20 110 (as represented by communication links 112 and 114). Each Web service 102
21 is illustrated as including one or more servers that execute software to handle
22 requests for particular services. Such services often maintain databases that store
23 information to be served back to requesters. Web services may be configured to
24 perform any one of a variety of different services. Examples of Web services
25 include login verification, notification, database storage, stock quoting, location



23

526

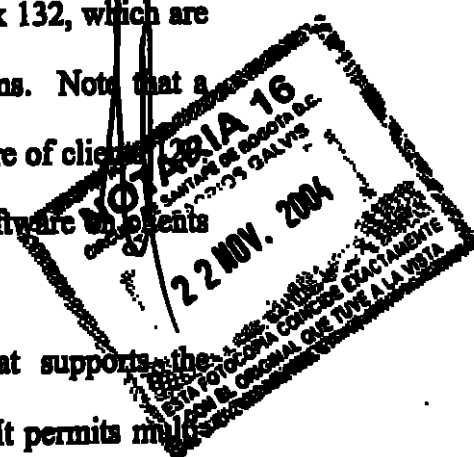
1 directories, mapping, music, electronic wallet, calendar/scheduler, telephone
2 listings, news and information, games, ticketing, and so on. The Web services can
3 be combined with each other and with other applications to build intelligent
4 interactive experiences.

5 The network environment 100 also includes representative client devices
6 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as
7 represented by communication link 122) and/or the Web application 110 (as
8 represented by communication links 124, 126, and 128). The clients may
9 communicate with one another using standard protocols as well, as represented by
10 an exemplary XML link 130 between clients 120(3) and 120(4).

11 The client devices, referenced generally as number 120, can be
12 implemented many different ways. Examples of possible client implementations
13 include, without limitation, portable computers, stationary computers, tablet PCs,
14 televisions/set-top boxes, wireless communication devices, personal digital
15 assistants, gaming consoles, printers, photocopiers, and other smart devices.

16 The Web application 110 is an application designed to run on the network
17 platform and may utilize the Web services 102 when handling and servicing
18 requests from clients 120. The Web application 110 is composed of one or more
19 software applications 130 that run atop a programming framework 132, which are
20 executing on one or more servers 134 or other computer systems. Note that a
21 portion of Web application 110 may actually reside on one or more of clients
22 Alternatively, Web application 110 may coordinate with other software on clients
23 120 to actually accomplish its tasks.

24 The programming framework 132 is the structure that supports the
25 applications and services developed by application developers. It permits multiple



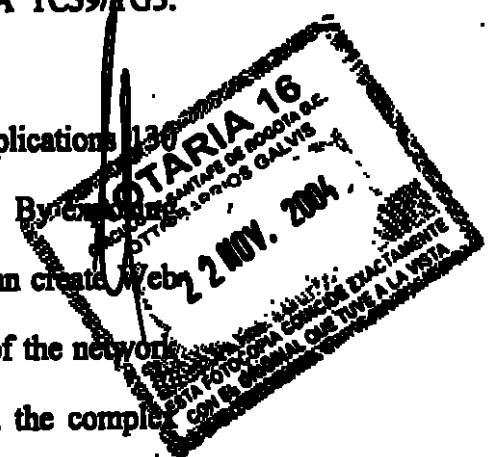
24

527

1 language development and seamless integration by supporting multiple languages.
2 It supports open protocols, such as SOAP, and encapsulates the underlying
3 operating system and object model services. The framework provides a robust and
4 secure execution environment for the multiple programming languages and offers
5 secure, integrated class libraries.

6 The framework 132 is a multi-tiered architecture that includes an
7 application program interface (API) layer 142, a common language runtime (CLR)
8 layer 144, and an operating system/services layer 146. This layered architecture
9 allows updates and modifications to various layers without impacting other
10 portions of the framework. A common language specification (CLS) 140 allows
11 designers of various languages to write code that is able to access underlying
12 library functionality. The specification 140 functions as a contract between
13 language designers and library designers that can be used to promote language
14 interoperability. By adhering to the CLS, libraries written in one language can be
15 directly accessible to code modules written in other languages to achieve seamless
16 integration between code modules written in one language and code modules
17 written in another language. One exemplary detailed implementation of a CLS is
18 described in an ECMA standard created by participants in ECMA TC39/TG3.
19 The reader is directed to the ECMA web site at www.ecma.ch.

20 The API layer 142 presents groups of functions that the applications 130
21 can call to access the resources and services provided by layer 146. By exposing
22 the API functions for a network platform, application developers can create Web
23 applications for distributed computing systems that make full use of the network
24 resources and other Web services, without needing to understand the complex
25 interworkings of how those network resources actually operate or are made



5

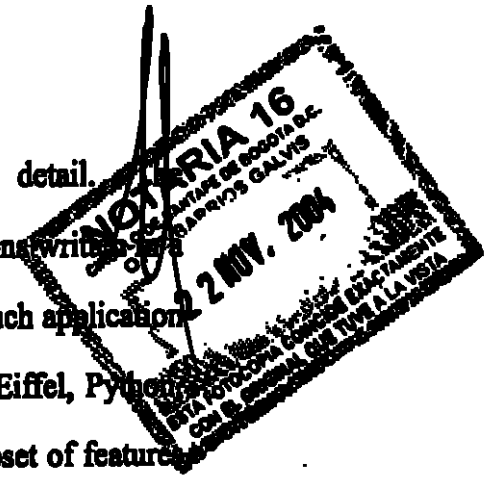
1 available. Moreover, the Web applications can be written in any number of
 2 programming languages, and translated into an intermediate language supported
 3 by the common language runtime 144 and included as part of the common
 4 language specification 140. In this way, the API layer 142 can provide methods
 5 for a wide and diverse variety of applications.

6 Additionally, the framework 132 can be configured to support API calls
 7 placed by remote applications executing remotely from the servers 134 that host
 8 the framework. Representative applications 148(1) and 148(2) residing on clients
 9 120(3) and 120(M), respectively, can use the API functions by making calls
 10 directly, or indirectly, to the API layer 142 over the network 104.

11 The framework may also be implemented at the clients. Client 120(3)
 12 represents the situation where a framework 150 is implemented at the client. This
 13 framework may be identical to server-based framework 132, or modified for client
 14 purposes. Alternatively, the client-based framework may be condensed in the
 15 event that the client is a limited or dedicated function device, such as a cellular
 16 phone, personal digital assistant, handheld computer, or other
 17 communication/computing device.

18 19 DEVELOPERS' PROGRAMMING FRAMEWORK

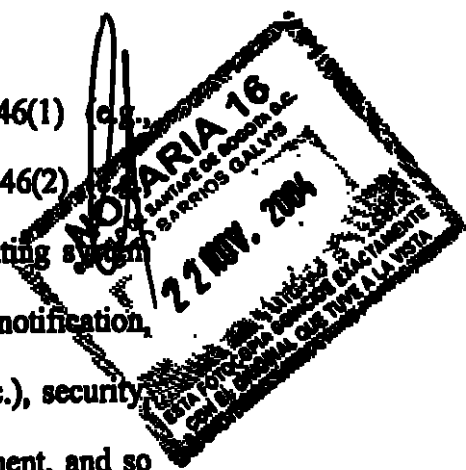
20 Fig. 2 shows the programming framework 132 in more detail.
 21 common language specification (CLS) layer 140 supports applications written in a
 22 variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such applications
 23 languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python
 24 and so on. The common language specification 140 specifies a subset of features
 25 or rules about features that, if followed, allow the various languages to



529

communicate. For example, some languages do not support a given type (e.g., an "int*" type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the "int[]" type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled "Method and System for Compiling Multiple Languages", filed 6/21/2000 (serial number 09/598,105) and "Unified Data Type System and Method" filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security, authentication, verification, processes and threads, memory management, and so



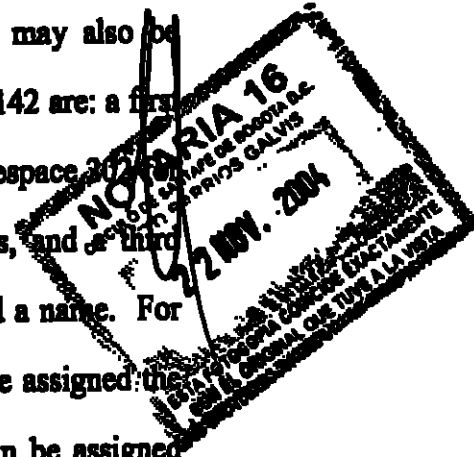
27

586
530

1 on. The object model services 146(2) provide interfacing with other objects to
2 perform various tasks. Calls made to the API layer 142 are handed to the common
3 language runtime layer 144 for local execution by the operating system 146(1)
4 and/or object model services 146(2).

5 The API 142 groups API functions into multiple namespaces. Namespaces
6 essentially define a collection of classes, interfaces, delegates, enumerations, and
7 structures, which are collectively called "types", that provide a specific set of
8 related functionality. A class represents managed heap allocated data that has
9 reference assignment semantics. A delegate is an object oriented function pointer.
10 An enumeration is a special kind of value type that represents named constants. A
11 structure represents static allocated data that has value assignment semantics. An
12 interface defines a contract that other types can implement.

13 By using namespaces, a designer can organize a set of types into a
14 hierarchical namespace. The designer is able to create multiple groups from the
15 set of types, with each group containing at least one type that exposes logically
16 related functionality. In the exemplary implementation, the API 142 is organized
17 to include three root namespaces. It should be noted that although only three root
18 namespaces are illustrated in Fig. 2, additional root namespaces may also be
19 included in API 142. The three root namespaces illustrated in API 142 are: a first
20 namespace 200 for a presentation subsystem (which includes a namespace 202 for
21 a user interface shell), a second namespace 204 for web services, and a third
22 namespace 206 for a file system. Each group can then be assigned a name. For
23 instance, types in the presentation subsystem namespace 200 can be assigned the
24 name "Windows", and types in the file system namespace 206 can be assigned
25 names "Storage". The named groups can be organized under a single "global



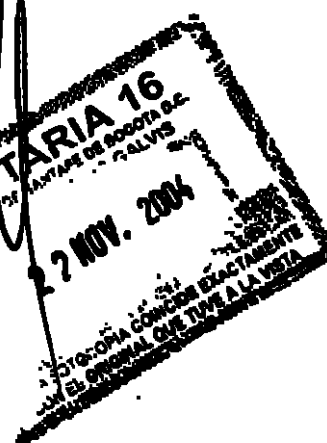
531

1 root" namespace for system level APIs, such as an overall System namespace. By
2 selecting and prefixing a top level identifier, the types in each group can be easily
3 referenced by a hierarchical name that includes the selected top level identifier
4 prefixed to the name of the group containing the type. For instance, types in the
5 file system namespace 206 can be referenced using the hierarchical name
6 "System.Storage". In this way, the individual namespaces 200, 204, and 206
7 become major branches off of the System namespace and can carry a designation
8 where the individual namespaces are prefixed with a designator, such as a
9 "System." prefix.

10 The presentation subsystem namespace 200 pertains to programming and
11 content development. It supplies types that allow for the generation of
12 applications, documents, media presentations and other content. For example,
13 presentation subsystem namespace 200 provides a programming model that allows
14 developers to obtain services from the operating system 146(1) and/or object
15 model services 146(2).

16 The shell namespace 202 pertains to user interface functionality. It supplies
17 types that allow developers to embed user interface functionality in their
18 applications, and further allows developers to extend the user interface
19 functionality.

20 The web services namespace 204 pertains to an infrastructure for enabling
21 creation of a wide variety of applications, e.g. applications as simple as a chat
22 application that operates between two peers on an intranet, and/or as complex as a
23 scalable Web service for millions of users. The described infrastructure is
24 advantageously highly variable in that one need only use those parts that are
25 appropriate to the complexity of a particular solution. The infrastructure provides



29

532

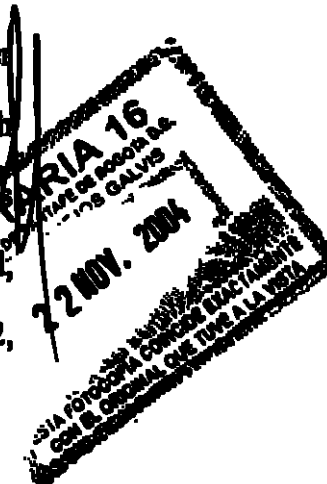
1 a foundation for building message-based applications of various scale and
2 complexity. The infrastructure or framework provides APIs for basic messaging,
3 secure messaging, reliable messaging and transacted messaging. In the
4 embodiment described below, the associated APIs have been factored into a
5 hierarchy of namespaces in a manner that has been carefully crafted to balance
6 utility, usability, extensibility and versionability.

7 The file system namespace 206 pertains to storage. It supplies types that
8 allow for information storage and retrieval.

9 In addition to the framework 132, programming tools 210 are provided to
10 assist the developer in building Web services and/or applications. One example of
11 the programming tools 210 is Visual Studio™, a multi-language suite of
12 programming tools offered by Microsoft Corporation.

14 ROOT API NAMESPACES

15 Fig. 3 shows a portion of the presentation subsystem 200 in more detail. In
16 one embodiment, the namespaces are identified according to a hierarchical naming
17 convention in which strings of names are concatenated with periods. For instance,
18 the presentation subsystem namespace 200 is identified by the root name
19 "System.Windows". Within the "System.Windows" namespace is another
20 namespace for various controls, identified as "System.Windows.Controls", which
21 further identifies another namespace for primitives (not shown) known as
22 "System.Windows.Controls.Primitives". With this naming convention in mind,
23 the following provides a general overview of selected namespaces of the API 142,
24 although other naming conventions could be used with equal effect.



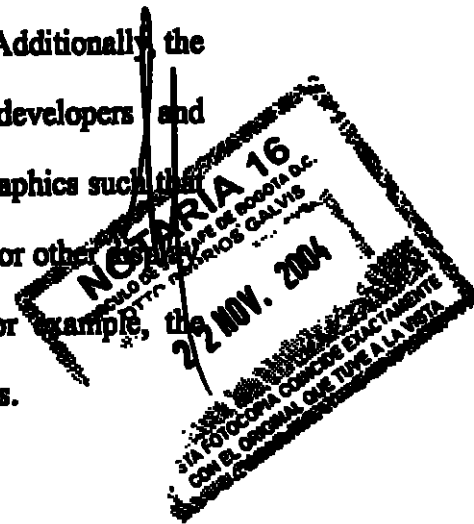
30

533

As shown in Fig. 3, the presentation subsystem 200 includes multiple namespaces. The namespaces shown in Fig. 3 represent a particular embodiment of the presentation subsystem 200. Other embodiments of the presentation subsystem 200 may include one or more additional namespaces or may omit one or more of the namespaces shown in Fig. 3.

The presentation subsystem 200 is the root namespace for much of the presentation functionality of the API 142. A controls namespace 310 includes controls used to build a display of information, such as a user interface, and classes that allow a user to interact with an application. Example controls include "Button" that creates a button on the display, "RadioButton" that generates a radio-style button on the display, "Menu" that creates a menu on the display, "ToolBar" that creates a toolbar on the display, "Image" that generates an image on the display and "TreeView" that creates a hierarchical view of information.

Certain controls are created by nesting and arranging multiple elements. The controls have a logical model that hides the elements used to create the controls, thereby simplifying the programming model. The controls can be styled and themed by a developer or a user (e.g., by customizing the appearance and behavior of user interface buttons). Some controls have addressable components that allow an individual to adjust the style of individual controls. Additionally, the controls can be sub-classed and extended by application developers and component developers. The controls are rendered using vector graphics such that they can be resized to fit the requirements of a particular interface or other. The controls are capable of utilizing animation to enhance, for example, the interactive feel of a user interface and to show actions and reactions.



31

534

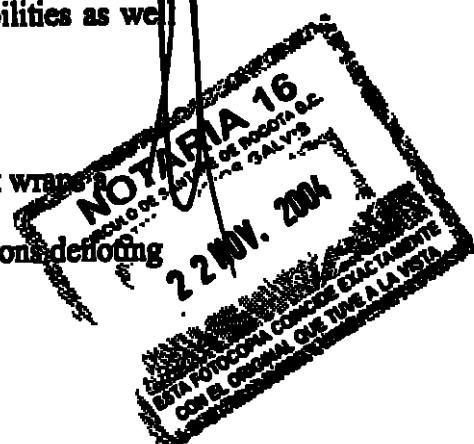
Controls Namespace

The controls namespace 310 includes one or more panels, which are controls that measure and arrange their children (e.g., nested elements). For example, a "DockPanel" panel arranges children by docking each child to the top, left, bottom or right side of the display, and fills-in the remaining space with other data. For example, a particular panel may dock menus and toolbars to the top of the display, a status bar to the bottom of the display, a folder list to the left side of the display, and fills the rest of the space with a list of messages.

As mentioned above, System.Windows.Controls.Primitives is a namespace that includes multiple controls that are components typically used by developers of the controls in the System.Windows.Controls namespace and by developers creating their own controls. Examples of these components include "Thumb and RepeatButton". "ScrollBar", another component, is created using four repeat buttons (one for "line up", one for "line down", one for "page up", and one for "page down") and a "Thumb" for dragging the current view to another location in the document. In another example, "ScrollViewer" is a control created using two "ScrollBars" and one "ScrollArea" to provide a scrollable area.

The following list contains example classes exposed by the System.Windows.Controls namespace. These classes allow a user to interact with, for example, an application through various input and output capabilities as well as additional display capabilities.

- **AccessKey** – AccessKey is a FrameworkElement element that wraps a character, indicating that it is to receive keyboard cue decorations denoting



32

535

the character as a keyboard mnemonic. By default, the keyboard cue decoration is an underline.

- **Audio** – Audio Element.
- **Border** – Draws a border, background, or both around another element.
- **Button** – Represents the standard button component that inherently reacts to the Click event.
- **Canvas** – Defines an area within which a user can explicitly position child elements by coordinates relative to the Canvas area.
- **CheckBox** – Use a CheckBox to give the user an option, such as true/false. CheckBox allows the user to choose from a list of options. CheckBox controls let the user pick a combination of options.
- **CheckedChangedEventArgs** – This CheckedChangedEventArgs class contains additional information about the CheckedChangedEvent event.
- **CheckStateChangedEventArgs** – This CheckStateChangedEventArgs class contains additional information about the CheckStateChangedEvent event.
- **ClickEventArgs** – Contains information about the Click event.
- **ColumnStyle** – Represents a changeable ColumnStyle object.
- **ColumnStyles** – Changeable pattern IList object that is a collection of Changeable elements.
- **ComboBox** – ComboBox control.

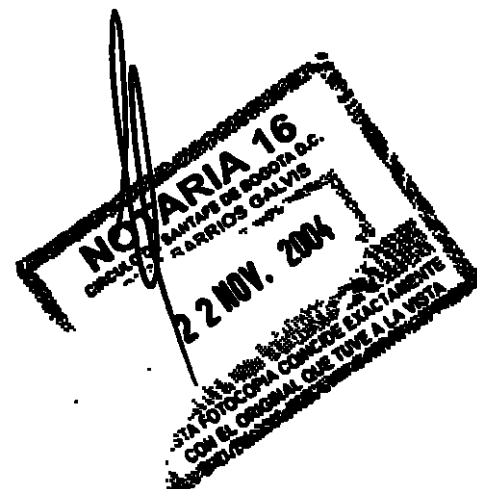


33

536

1 **ComboBoxItem** – Control that implements a selectable item inside a
2 **ComboBox**.

- 3 • **ContactPickerDialog** – Allows a user to select one or more contacts.
- 4 -----
- 5 • **ContactPropertyRequest** – Allows an application to request information
6 about a contact property through a **ContactPickerDialog**. This class cannot
7 be inherited.
- 8 • **ContactPropertyRequest Collection** – Represents a collection of
9 **ContactPropertyRequest** objects.
- 10 • **ContactSelection** – Information about a selected contact from Microsoft®
11 Windows® File System, code-named "WinFS" or Microsoft Active
12 Directory®.
- 13 • **ContactSelectionCollection** – Represents a collection of **ContactSelection**
14 objects.
- 15 • **ContactTextBox** – An edit control that supports picking contacts or
16 properties of contacts.
- 17 • **ContactTextBoxSelectionChangedEventArgs** – Arguments for the
18 **ContactTextBoxSelectionChanged** event.
- 19 • **ContactTextBoxTextChangedEventArgs** – Arguments for the
20 **ContactTextBoxTextChanged** event.
- 21 • **ContactTextBoxTextResolvedEventArgs** – Arguments for the
22 **ContactTextBoxTextResolved** event.
- 23 • **ContactTextBoxTextResolvedEventArgs** – Arguments for the
24 **TextResolvedToContact** event.
- 25

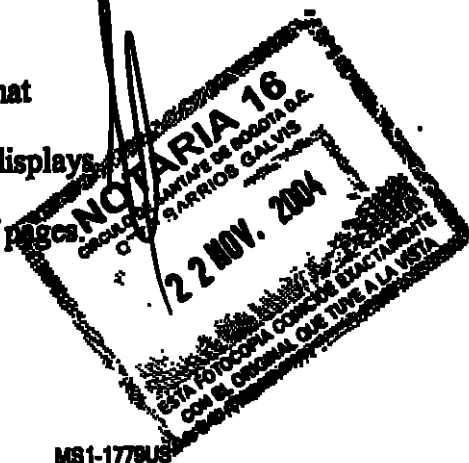


34

537

1 ContentChangedEventArgs – The event arguments for
2 ContentChangedEvent.

- 3 • ContentControl – The base class for all controls with a single piece of
4 content.
- 5
- 6 • ContentPresenter – ContentPresenter is used within the style of a content
7 control to denote the place in the control's visual tree (chrome template)
8 where the content is to be added.
- 9 • ContextMenu – Control that defines a menu of choices for users to invoke.
- 10
- 11 • ContextMenuEventArgs – The data sent on a ContextMenuEvent.
- 12
- 13 • Control – Represents the base class for all user-interactive elements. This
14 class provides a base set of properties for its subclasses.
- 15
- 16 • Decorator – Base class for elements that apply effects onto or around a
17 single child element, such as Border.
- 18
- 19 • DockPanel – Defines an area within which you can arrange child elements
20 either horizontally or vertically, relative to each other.
- 21
- 22 • DragDeltaEventArgs – This DragDeltaEventArgs class contains additional
23 information about the DragDeltaEvent event.
- 24
- 25 • FixedPanel – FixedPanel is the root element used in fixed-format
documents to contain fixed pages for pagination. FixedPanel displays
paginated content one page at a time or as a scrollable stack of pages.

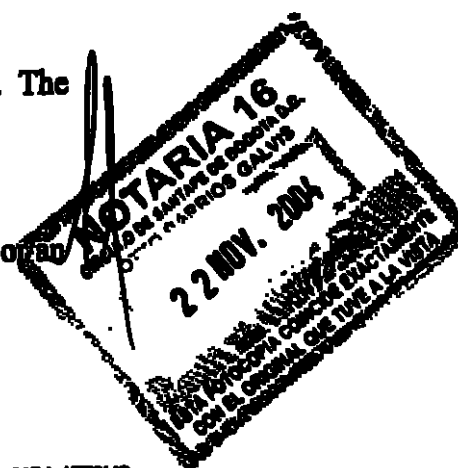


35

538

FlowPanel – FlowPanel is used to break, wrap, and align content that exceeds the length of a single line. FlowPanel provides line-breaking and alignment properties that can be used when the flow of the container's content, Text for example, is likely to exceed the length of a single line.

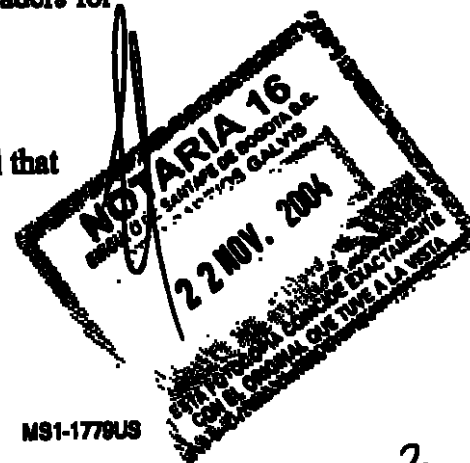
- **Frame** – An area that can load the contents of another markup tree.
- **Generator** – Generator is the object that generates a UI on behalf of an ItemsControl, working under the supervision of a GeneratorFactory.
- **GeneratorFactory** – A GeneratorFactory is responsible for generating the UI on behalf of an ItemsControl. It maintains the association between the items in the control's ItemsCollection (flattened view) and the corresponding UIElements. The control's item-container can ask the factory for a Generator, which does the actual generation of UI.
- **GridPanel** – Defines a grid area consisting of columns and rows.
- **HeaderItemsControl** – The base class for all controls that contain multiple items and have a header.
- **HorizontalScrollBar** – The Horizontal ScrollBar class.
- **HorizontalSlider** – The Horizontal Slider class.
- **HyperLink** – The HyperLink class implements navigation control. The default presenter is TextPresenter.
- **Image** – Provides an easy way to include an image in a document or an application.



36

1 IncludeContactEventArgs – Arguments passed to handlers of the
2 ContactPickerDialog.IncludeContact event.

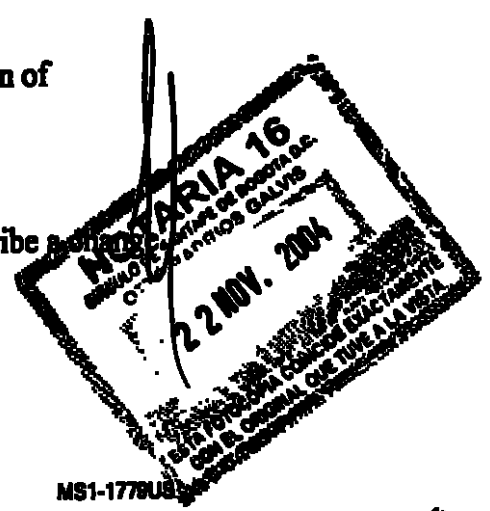
- 3 • ItemCollection – Maintains a collection of discrete items within a control.
4 Provides methods and properties that enable changing the collection
5 contents and obtaining data about the contents.
- 6
- 7 • ItemsChangedEventArgs – The ItemsChanged event is raised by a
8 GeneratorFactory to inform layouts that the items collection has changed.
- 9 • ItemsControl – The base class for all controls that have multiple children.
- 10
- 11 • ItemsView – ItemsView provides a flattened view of an ItemCollection.
- 12
- 13 • KeyboardNavigation – KeyboardNavigation class provide methods for
14 logical (Tab) and directional (arrow) navigation between focusable
15 controls.
- 16 • ListBox – Control that implements a list of selectable items.
- 17 • ListItem – Control that implements a selectable item inside a ListBox.
- 18 • Menu – Control that defines a menu of choices for users to invoke.
- 19 • MenuItem – A child item of Menu. MenuItems can be selected to invoke
20 commands. MenuItems can be separators. MenuItems can be headers for
21 submenus. MenuItems can be checked or unchecked.
- 22
- 23 • PageViewer – Represents a document-viewing composite control that
24 contains a pagination control, a toolbar, and a page bar control.
- 25



530
540

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

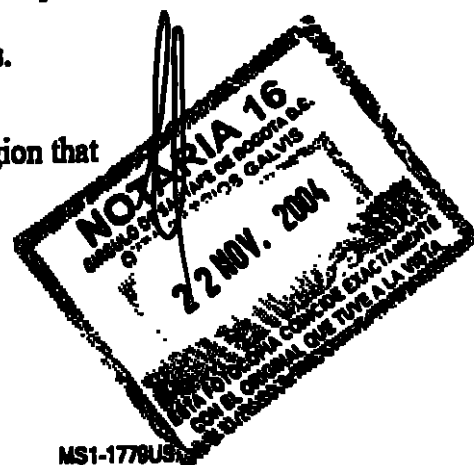
- PaginationCompleteEventArgs – The event arguments for the
PaginationCompleteEvent.
- PaginationProgressEventArgs – The event arguments for the
PaginationProgressEvent.
 - Pane – Provides a way to define window properties in a markup language
(e.g., "XAML") without launching a new window.
 - Panel – Provides a base class for all Panel elements. In order to instantiate
a Panel element, use the derived concrete class.
 - RadioButton – RadioButton implements an option button with two states:
true or false.
 - RadioButtonList – This control serves as a grouping control for
RadioButtons and is the piece that handles RadioButton mutual exclusivity.
The RadioButtonList inherits from Selector. The RadioButtonList is
essentially a Single SelectionMode Selector and the concept of Selection
(from Selector) is keyed off of the Checked property of the RadioButton it
is grouping.
 - RowStyle – Changeable pattern Changeable elements.
 - RowStyles – Changeable pattern IList object that is a collection of
Changeable elements.
 - ScrollChangeEventArgs – The ScrollChangeEventArgs describe a change
in scrolling state.



1 ScrollViewer –

2 SelectedItemsCollection – A container for the selected items in a Selector.

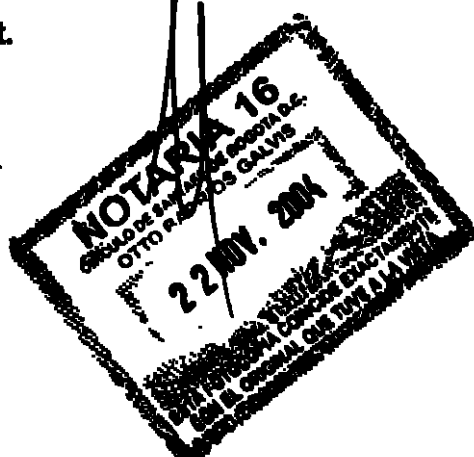
- 3
- 4 • SelectionChangedEventArgs – The inputs to a selection changed event handler.
- 5
- 6 • SimpleText – SimpleText is a lightweight, multi-line, single-format text
- 7 element intended for use in user interface (UI) scenarios. SimpleText
- 8 exposes several of the same formatting properties as Text and can often be
- 9 used for a performance gain at the cost of some versatility.
- 10
- 11 • StyleSelector – StyleSelector allows the app writer to provide custom style
- 12 selection logic. For example, with a class Bug as the Content, use a
- 13 particular style for Pri1 bugs and a different style for Pri2 bugs. An
- 14 application writer can override the SelectStyle method in a derived selector
- 15 class and assign an instance of this class to the StyleSelector property on
- 16 ContentPresenter class.
- 17 • Text – Represents a Text control that enables rendering of multiple formats
- 18 of Text. Text is best used within an application UI; more advanced text
- 19 scenarios benefit from the additional feature set of TextPanel. In most
- 20 cases where relatively simple text support is required, Text is the preferred
- 21 element because of its lightweight nature and range of features.
- 22 • TextBox – Represents the control that provides an editable region that
- 23 accepts text input.
- 24
- 25



542

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

- **TextChangedEventArgs** – The **TextChangedEventArgs** class represents a type of **RoutedEventArgs** that are relevant to events raised by **TextRange.SetText()**.
- **TextPanel** – Formats, sizes, and draws text. **TextPanel** supports multiple lines of text and multiple text formats.
- **ToolTip** – A control to display information when the user hovers over a control.
- **ToolTipEventArgs** – The data sent on a **ToolTipEvent**.
- **TransformDecorator** – **TransformDecorator** contains a child and applies a specified transform to it. **TransformDecorator** implements logic to measure and arrange the child in its local (pre-transform) coordinates such that after the transform, the child fits tightly within the decorator's space and uses maximal area. The child therefore needs to have no knowledge that a transform has been applied to it.
- **UIElementCollection** – A **UIElementCollection** is a ordered collection of **UIElements**.
- **ValueChangedEventArgs** – This **ValueChangedEventArgs** class contains additional information about the **ValueChangedEvent** event.
- **VerticalScrollBar** – The **Vertical ScrollBar** class.
- **VerticalSlider** – The **Vertical Slider** class.



46

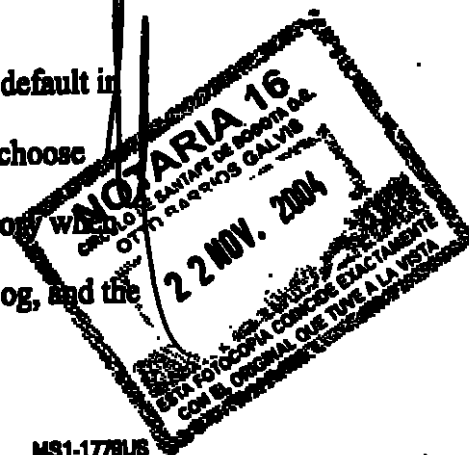
543

1 Video – Plays a streaming video or audio file in a specified rectangle within
2 the current user coordinate system.

- 3 • VisibleChangedEventArgs – The VisibleChangedEventArgs class contains
4 additional information about the VisibleChangedEvent event.

5
6
7 The System.Windows.Controls namespace also contains various
8 enumerations. The following list contains example enumerations associated with
9 the System.Windows.Controls namespace.

- 10 • CharacterCase – Specifies the case of characters in a TextBox control when
11 the text is typed.
- 12
13 • CheckState – Specifies the state of a control, such as a check box, that can
14 be checked, unchecked, or set to an indeterminate state.
- 15
16 • ClickMode – Specifies when the Click event should fire.
- 17
18 • ContactControlPropertyPosition – Controls the position and display of the
19 property of the contact.
- 20
21 • ContactPickerDialogLayout – Specifies how the ContactPickerDialog
22 should display selected properties.
- 23
24 • ContactPropertyCategory – Specifies which value to treat as the default in
25 the case where a property has multiple values for the user could choose
from. For example, if "Work" is specified as the preferred category when
requesting a phone number property from the ContactPickerDialog, and the



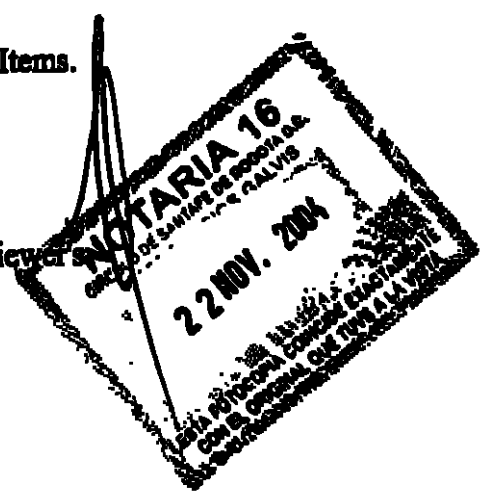
41

544

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

user selects a contact with both a work and home phone number, the work phone number appears as the default selection. The user can then use the UI to choose the home phone number instead.

- **ContactPropertyType** – Specifies a property of a contact that the **ContactPickerDialog** can ask the user for.
- **ContactType** – Specifies which contact types to display in the **ContactPickerDialog**.
- **Direction** – This enumeration is used by the **GeneratorFactory** and **Generator** to specify the direction in which the generator produces UI.
- **Dock** – Specifies the Dock position of a child element within a **DockPanel**.
- **GeneratorStatus** – This enumeration is used by the **GeneratorFactory** to indicate its status.
- **KeyNavigationMode** – The type of **TabNavigation** property specify how the container will move the focus when Tab navigation occurs.
- **MenuItemBehavior** – Defines the different behaviors that a **MenuItem** could have.
- **MenuItemType** – Defines the different placement types of **MenuItems**.
- **Orientation** – Slider orientation types.
- **PageViewerFit** – Selects how pages should be fit into the **PageViewer** Client area.



42

545

1 **PageViewerMode** – Selects the current PageViewer mode Reflected in the
2 mode dropdown.

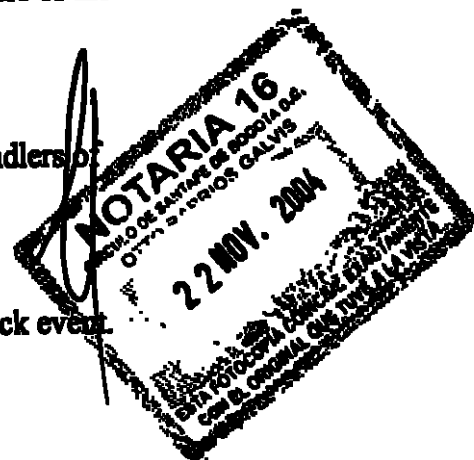
- 3 • **ScrollerVisibility** – **ScrollerVisibilty** defines the visibility behavior of a
4 scrollbar.
- 5 • **SelectionMode** – Specifies the selection behavior for the ListBox.

6
7
8 "Position" is an example structure associated with the
9 System.Windows.Controls namespace. A user of the Generator describes
10 positions using this structure. For example: To start generating forward from the
11 beginning of the item list, specify position (-1, 0) and direction Forward. To start
12 generating backward from the end of the list, specify position (-1, 0) and direction
13 Backward. To generate the items after the element with index k, specify position
14 (k, 0) and direction Forward.

15
16
17 The following list contains example delegates associated with the
18 System.Windows.Controls namespace.

- 19 • **CheckedChangedEventHandler** – This delegate is used by handlers of the
20 **CheckedChangedEvent** event.
- 21 • **CheckStateChangedEventHandler** – This delegate is used by handlers of
22 the **CheckStateChangedEvent** event.

23 **ClickEventHandler** – Represents the methods that handle the Click event.

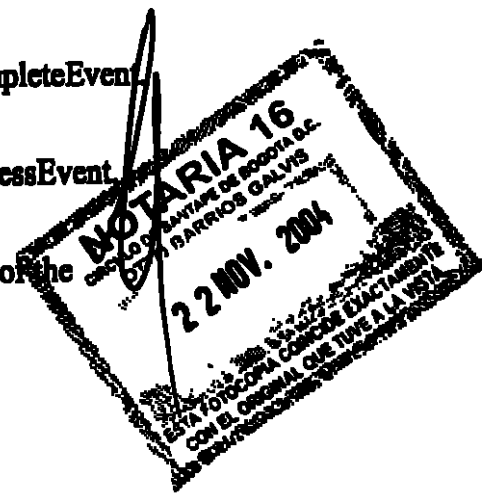


43

546

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

- ContactTextBoxSelectionChangedEventHandler** – A delegate handler for the **ContactTextBoxSelectionChanged** event.
- **ContactTextBoxTextChangedEventHandler** – A delegate handler for the **ContactTextBoxTextChanged** event.
- **ContactTextBoxTextResolvedEventHandler** – A delegate handler for the **TextResolvedToContact** event.
- **ContentChangedDelegate** – Delegate for the **ContentChangedEvent**.
- **ContextMenuEventHandler** - The callback type for handling a **ContextMenuEvent**.
- **DragDeltaEventHandler** – This delegate is used by handlers of the **DragDeltaEvent** event.
- **IncludeContactEventHandler** – Handler for **ContactPickerDialog.IncludeContact** event.
- **ItemsChangedEventHandler** – The delegate to use for handlers that receive **ItemsChangedEventArgs**.
- **OpenedEventHandler** – Handler for **ContactPickerDialog.Opened** event.
- **PaginationCompleteDelegate** – Delegate for the **PaginationCompleteEvent**.
- **PaginationProgressDelegate** – Delegate for the **PaginationProgressEvent**.
- **ScrollChangeEventHandler** – This delegate is used by handlers of the **ScrollChangeEvent** event.



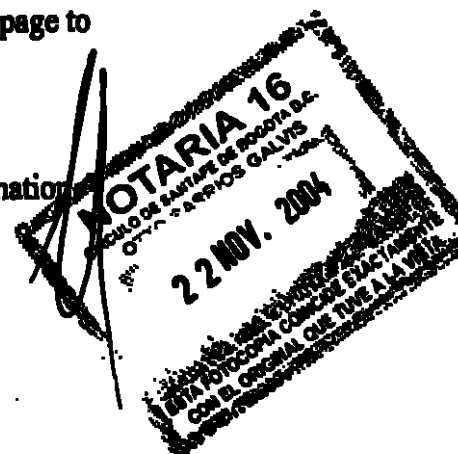
54

547

- **SelectionChangedEventHandler** – The delegate type for handling a selection changed event.
- **TextChangedEventHandler** – The delegate to use for handlers that receive **TextChangedEventArgs**.
- **ToolTipEventHandler** – The callback type for handling a **ToolTipEvent**.
- **ValueChangedEventHandler** – This delegate is used by handlers of the **ValueChangedEvent** event.
- **VisibleChangedEventHandler** – This delegate is used by handlers of the **VisibleChangedEvent** event.

Another namespace, **System.Windows.Controls.Atoms**, is a sub-namespace of the **System.Windows.Controls** namespace. **System.Windows.Controls.Atoms** includes associated controls, event arguments and event handlers. The following list contains example classes associated with the **System.Windows.Controls.Atoms** namespace.

- **PageBar** – Represents a scrollable pagination control.
- **PageElement** – Renders a specific page of paginated content. The page to be rendered is specified by the **PageSource** property.
- **PageHoveredEventArgs** – **PageHoveredEventArgs** provides information about where the mouse pointer is hovering.



25

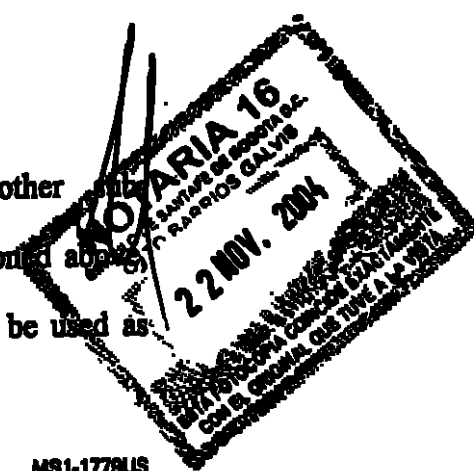
PageScrolledEventArgs – The **PageScrolledEventArgs** contains info pertaining to the **PageScrolled** Event.

- **PageSelectedEventArgs** – The **PageSelectedEvent** is fired when a new row/column range selection is made.
- **PageSelector** – **PageSelector**: Allows the user to select a range of rows/columns of pages to be displayed.
- **PageSource** – Identifies the source of the content to be paginated. It also provides properties and methods for formatting paginated content.

The following list contains example delegates associated with the **System.Windows.Controls.Atoms** namespace.

- **PageHoveredEventHandler** – This delegate is used by handlers of the **PageHoveredEvent** event.
- **PageScrolledEventHandler** – This delegate is used by handlers of the **PageHovered** event.
- **PageSelectedEventHandler** – This delegate is used by handlers of the **PageSelectedEvent** event.

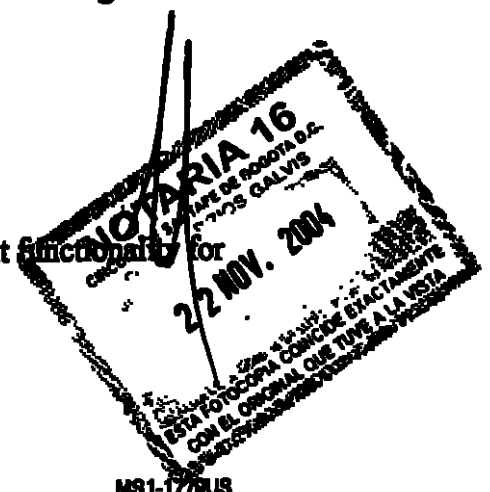
A **System.Windows.Controls.Primitives** namespace is another namespace of the **System.Windows.Controls** namespace. As mentioned above, the **Primitives** sub-namespace includes controls that are intended to be used as



549

1 primitives by other more complex controls. The following list contains example
2 classes associated with the System.Windows.Controls.Primitives namespace.

- 3 • **ButtonBase** – When overridden in a derived class, defines the relevant
4 events and properties, and provides handlers for the relevant input events.
- 5 • **Popup** – A control that creates a fly-out window that contains content.
- 6 • **RangeBase** – Represents the base class for elements that have a specific
7 range. Examples of such elements are scroll bars and progress bars. This
8 class defines the relevant events and properties, and provides handlers for
9 the events.
- 10 • **RepeatButton** – RepeatButton control adds repeating semantics of when the
11 Click event occurs.
- 12 • **ScrollArea** – ScrollArea is the effective element for scrolling. It contains
13 content that it clips and provides properties to expose the content's offset
14 and extent. It also provides default input handling such that scrolling can
15 be driven programatically or via keyboard or mouse wheel.
- 16 • **ScrollBar** – The ScrollBar class.
- 17 • **Selector** – The base class for controls that select items from among their
18 children.
- 19 • **Slider** – The Slider class.
- 20 • **Thumb** – The thumb control enables basic drag-movement functionality for
21 scrollbars and window resizing widgets.
- 22
- 23
- 24
- 25



47

340
550

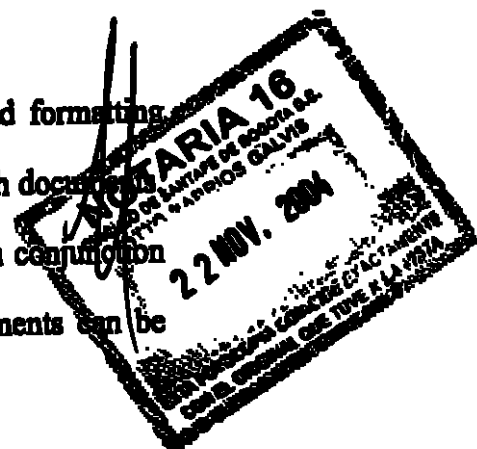
1
2 "IEnsureVisible" is an example interface associated with the
3 System.Windows.Controls.Primitives namespace. IEnsureVisible is implemented
4 on a visual to scroll/move a child visual into view.

5
6 The following list contains example enumerations associated with the
7 System.Windows.Controls.Primitives namespace.

- 8 • ArrowButtonStates –
- 9 • CloseModeType – Describes how a popup should behave to various mouse
- 10 events.
- 11
- 12 • Part – The Part enumeration is used to indicate the semantic use of the
- 13 controls that make up the scroll bar.
- 14 • PartStates – ScrollBar Part States.
- 15
- 16 • PlacementType – Describes where a popup should be placed on screen.
- 17
- 18 • SizeBoxStates –

19
20 Documents Namespace

21 A documents namespace 312 is a collection of semantic and formatting
22 elements that are used to create richly formatted and semantically rich documents.
23 In one embodiment, an "element" is a class that is primarily used in conjunction
24 with a hierarchy of elements (referred to as a "tree"). These elements can be
25



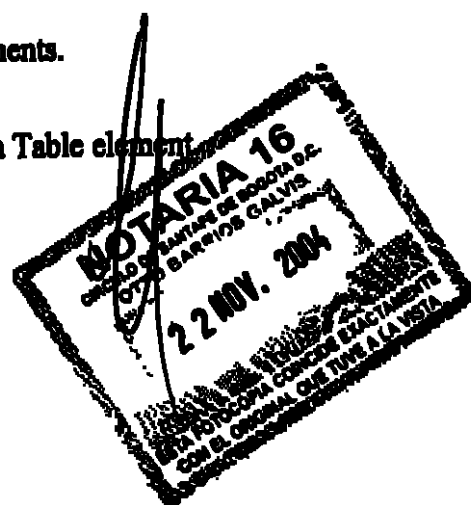
48

551

1 interactive (e.g., receiving user input via keyboard, mouse or other input device),
2 can render images or objects, and can assist with the arrangement of other
3 elements. Example elements include a "Block" element that implements a generic
4 block, a "Body" element that represents content that includes the body of a table, a
5 "Cell" element that contains tabular data within a table, a "Header" element that
6 represents the content included in the header of a table, and a "PageBreak"
7 element that is used to break content across multiple pages.

8 The following list contains example classes exposed by the
9 System.Windows.Documents namespace.

- 10 • AdaptiveMetricsContext - AdaptiveMetricsContext provides the root
11 element for adaptive-flow-format documents. Once a child panel is
12 encapsulated in an AdaptiveMetricsContext element, the content of the
13 panel is processed by the Reading Metrics Engine (RME). The size of the
14 child panel is used to calculate the number and size of any columns as well
15 as optimum font sizes and line heights.
- 16 • Block - Implements a generic block element that does not induce default
17 rendering behavior.
- 18 • BlockElement - Implements a base class for all Block elements.
- 19 • Body - Represents the content that comprises the body of a Table element.
- 20 • Bold - Implements a Bold element derived from Inline.
- 21
- 22
- 23
- 24
- 25



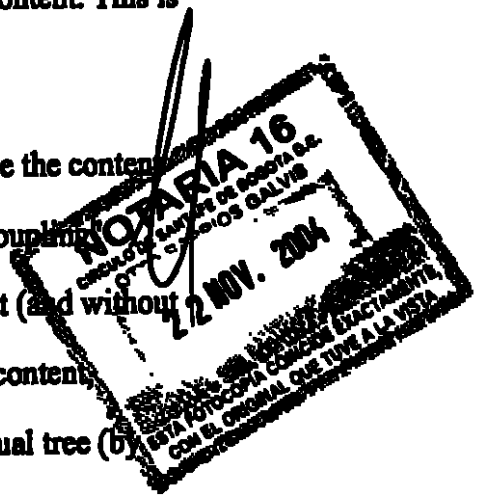
29

552

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

BreakRecord - Stores information necessary to continue formatting paginated content across page breaks. Inherit from this class to provide pagination support. This is an abstract class.

- **Cell** - Cells contain tabular data within a Table. Cell elements are contained within a Row.
- **CellCollection** - Ordered collection of table cells.
- **Column** - The Column element is used to apportion the contents of a GridPanel or Table.
- **ColumnCollection** - A ColumnCollection is an ordered collection of Columns.
- **ColumnResult** - Represents a column's view-related information.
- **ContainerParagraphResult** - Provides access to calculated layout parameters for a Paragraph object which contains only other Paragraph objects.
- **ContentPosition** - Represents the position of content within a paragraph. Inherit from this class to describe the position of associated content. This is an abstract class.
- **Document** - The purpose of the Document class is to decouple the content of a document from the UI "chrome" that surrounds it. "Decoupling" means that you can author a document without thinking about (and without committing to) its UI. The Document class holds document content, typically a TextPanel or a FixedPanel and its children. A visual tree (by

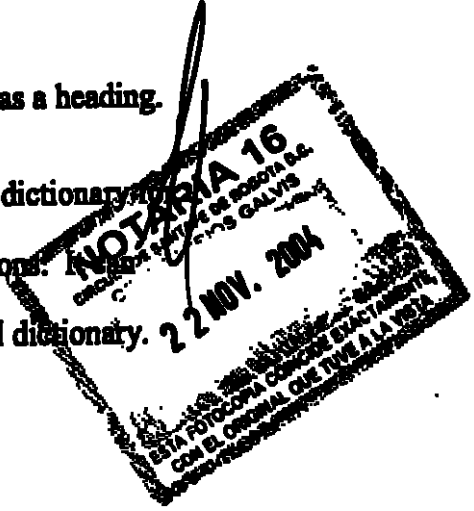


50

553

default, a PageViewer) is associated with this element through the WPP control styling mechanism.

- **DocumentPage** - Represents layout information for a control associated with a page of a document subject to pagination. Inherit from this class to implement to describe the layout information for these controls. This is an abstract class.
- **DocumentPageParagraphResult** - Provides access to calculated layout parameters for objects affected by pagination.
- **FindEngine** - Base class for find algorithms.
- **FindEngineFactory** - Find algorithms factory.
- **FixedPage** - Provides access to a single page of content within a fixed-format layout document.
- **Footer** - Represents the content that comprises the footer of a Table element.
- **Header** - Represents the content that comprises the header of a Table element.
- **Heading** - Implements a block-level element that renders text as a heading.
- **HyphenationDictionary** - HyphenationDictionary represents a dictionary the purpose of providing hyphenation support within applications. It can contain both an inline dictionary and a reference to an external dictionary.

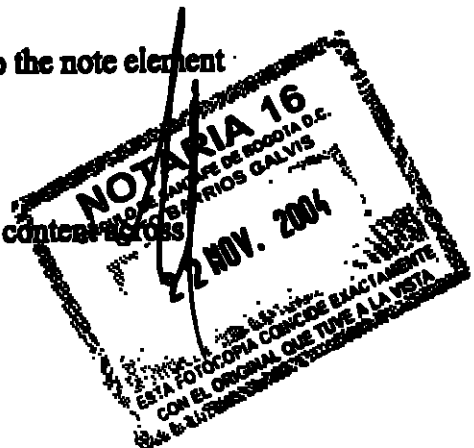


21

554

The inline dictionary has higher priority and will be applied before entries in the external dictionary.

- **Hyphenator** - The Hyphenator object maintains reference to hyphenation data within a HyphenationDictionary and also performs hyphenation.
- **Inline** - Implements a generic Inline element that does not induce any default rendering behavior.
- **InlineElement** - Implements a generic inline element as base class for all inline elements.
- **Italic** - Implements an Italic element derived from Inline.
- **LineBreak** - Represents a markup element that forces a line break.
- **LineResult** - Provides access to calculated information of a line of text.
- **List** - Implements a List element. Lists are block-level elements designed to be formatted with markers such as bullets or numbering.
- **ListElementItem** - Implements a ListElementItem, which supports markers such as bullets or numbering.
- **Note** - Implements a Note element, which is analagous to the note element in HTML.
- **PageBreak** - Represents a markup element used to break content across various pages.



52

555

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

- PageDescriptor - Implements PageDescriptor, which stores information necessary to create paginated layout.
- Paragraph - Implements a block-level element used to render text in a paragraph. Rendering behavior is analagous to that of the paragraph element in HTML.
- ParagraphResult - Provides access to calculated layout parameters for a Paragraph object.
- Row - Defines a row within a GridPanel or Table element.
- RowCollection - RowCollection represents an ordered collection of Rows.
- RowGroup - Specifies property defaults for a group of rows in a Table or GridPanel.
- Section - Implements a generic container element. Rendering behavior is analagous to the div element in HTML.
- SmallCaps - Implements an inline SmallCaps element. SmallCaps are typographic forms that render as small capital versions of letters for emphasis, as in a title.
- Subscript - Represents an inline Subscript element. Subscript characters are written immediately below, below and to the left, or below and to the right of other characters.

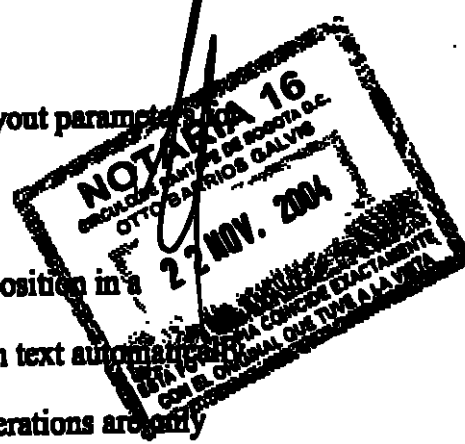


53

556

Superscript - Represents an inline Superscript element. Superscript characters are typically letters or numbers and render immediately above, above and to the left, or above and to the right of other characters.

- Table - Table is used to display complex data in tabular form using a markup language (e.g., "XAML").
- TextArray - Base API for text access and manipulation.
- TextChangedEventArgs - The TextChangedEventArgs defines the event arguments sent when a TextArray is changed.
- TextElement - TextElement provides TextRange facilities for the TextTree. It is an immutable, continuous TextRange with fixed endpoints. It provides ContentElement Input, Focus and Eventing support. It also provides DependencyObject property support.
- TextNavigator - This can enumerate text content. Implements a movable TextPosition. It can move by text run or be positioned at a know location in text.
- TextParagraphResult - Provides access to calculated layout parameters for text, including floated objects and figures.
- TextPosition - This is an object representing a certain position in a TextArray. A compact object representing a position in text automatically maintains position when text changes. Comparison operations are only applicable to positions within same TextArray (same Context) TextPosition can be static or movable. IsChangeable property tells the kind of position.



54

55

1 **TextRange** - **TextRange** is an abstract class providing generic association of
2 zero or more subranges with properties. Subrange manipulation is defined
3 on derived classes.

- 4 • **TextRangeMovable** - **TextRangeMovable** is an abstract class for movable
5 **TextRanges**. It adds the ability to move the start and end points based on
6 **TextUnits**.
- 7 • **TextTreeChangedEventArgs** - The **TextTreeChangedEventArgs** defines the
8 event arguments sent when a **TextArray** is changed.
- 9 • **TextTreeDumper** - **TreeDumper** is a tree test class that is public due to
10 packaging issues.
- 11 • **TextTreeNavigator** - This is an object representing a certain moveable
12 position in a **TextTree**. It is a specific implementation of **TextNavigator** for
13 use only in the **TextTree**.
- 14 • **TextTreePosition** - This is an object representing a certain immutable
15 position in a **TextTree**. It is a specific implementation of **TextPosition** for
16 use only in the **TextTree**.
- 17 • **TextTreeRange** - Provides **TextRange** facilities for the **TextTree**. It is a
18 mutable, continuous **TextRange** with movable endpoints.
- 19 • **TextTreeRangeContentEnumerator** - Enumerator on object children directly
20 under a **TextTreeRange**.
- 21 • **TextTreeRangeContentEnumerator** - Enumerator on object children directly
22 under a **TextTreeRange**.
- 23 • **TextTreeRangeContentEnumerator** - Enumerator on object children directly
24 under a **TextTreeRange**.

25 **TextUnit** - Extensible unit of text navigation.



55

558

1 **TextUnits** - Commonly used text units for **TextPosition** and **TextRange**.

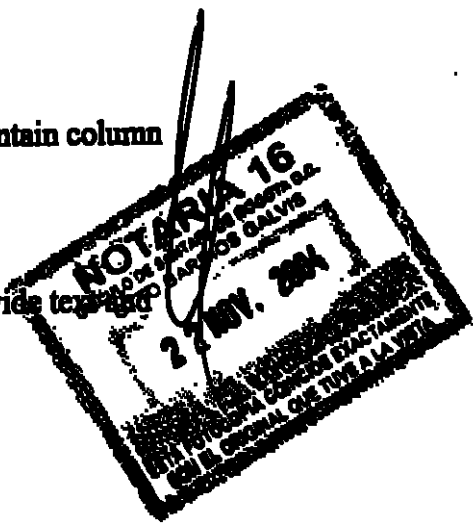
2 **Typography** - Provides access to a rich set of **OpenType** typography
3 properties.

- 4 -----
- 5 • **UIElementParagraphResult** - The **ParagraphResult** for a paragraph which is
6 composed entirely of a **UIElement**. Used for **Floater**s, **Figure**s and
7 embedded block level **UIElement**s.
 - 8 • **Underline** - Implements an **Underline** element derived from **InlineElement**.
- 9

10

11 The following list contains example interfaces associated with the
12 **System.Windows.Documents** namespace.

- 13 • **IDocumentContentHost** - Implement this interface on a content host so that
14 children of that host can notify the host when content is changing.
 - 15 • **IDocumentFormatter** - Implement this interface on an element to provide
16 support for document features such as pagination.
 - 17 • **ITextDocumentResult** - Implement this interface to maintain column
18 information for a document.
 - 19 • **ITextParagraphResult** - Implement this interface to provide text
20 positioning information for text paragraphs.
- 21
- 22
- 23
- 24
- 25



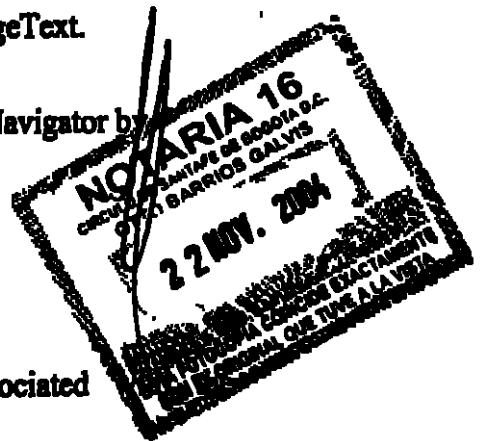
56

551

1 The following list contains example enumerations associated with the
2 System.Windows.Documents namespace.

- 3 • **ElementEdge** - This identifies the edge of an object where a **TextPosition** is
4 located.
- 5 • **FindAdvancedOptions** - The advanced search options used by
6 **FindAlgorithm** (search initialization) and
7 **TextRangeMovable/TextSelection** (simplified search execution) classes.
- 8 • **FindOptions** - The simplified search options used by **TextBox.Find**
9 methods.
- 10 • **LogicalDirection** - **LogicalDirection** defines a logical direction for
11 movement in text. It is also used to determine where a **TextPosition** will
12 move when content is inserted at the **TextPosition**.
- 13 • **TextArrayRunType** - This identifies the run where a **TextPosition** is
14 located, taking **LogicalDirection** into account.
- 15 • **TextChangeOptions** - Possible text changes for **CanChangeText**.
- 16 • **TextMoveOptions** - This controls the movement of **TextNavigator** by
17 specifying conditions to halt navigation.
- 18
- 19
- 20
- 21
- 22

23 The following list contains example delegates associated
24 System.Windows.Documents namespace.
25



57

560

ObjectCloneDelegate - Callback method to provide a clone or copy of a DependencyObject when a portion of a TextArray is being copied or moved.

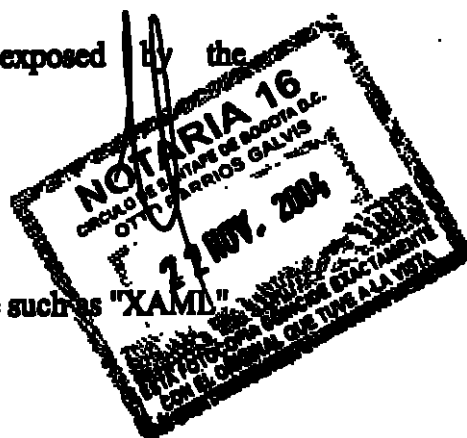
- TextChangedEventHandler - The TextChangedEventHandler delegate is called with TextChangedEventArgs every time content is added to or removed from the TextTree.

Shapes Namespace

A shapes namespace 314 is a collection of vector graphics elements that is used to create images and objects. The use of vector graphics elements allows the elements to be easily resized to fit the requirements of a particular interface or display device. Example elements include an "Ellipse" element that draws an ellipse, a "Line" element that draws a straight line between two points, a "Rectangle" element that draws a rectangle, and a "Polygon" element that draws a polygon as a connected series of lines that form a closed shape.

The following list contains example classes exposed by the System.Windows.Shapes namespace.

- Ellipse - Draws an ellipse.
- Glyphs - Represents a glyph shape in a markup language such as "XAML".
Glyphs are used to represent fonts.
- Line - Draws a straight line between two points.
- Path - Draws a series of connected lines and curves.



56

561

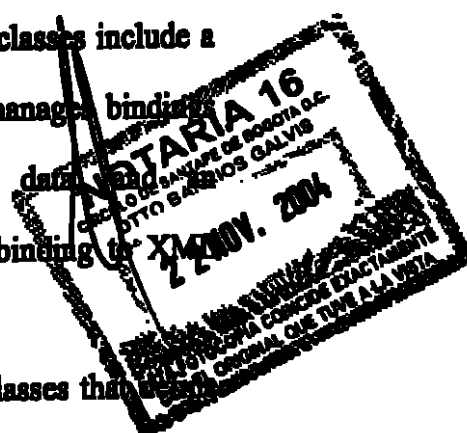
1 Polygon – Draws a polygon (a connected series of lines that forms a closed
2 shape).

- 3 • Polyline – Draws a series of connected straight lines.
- 4
- 5 • Rectangle – Draws a rectangle.
- 6
- 7 • Shape – An abstract class that provides base functionality for shape
8 elements, such as ellipse, polygon and rectangle.
- 9

10 Data Namespace

11 A data namespace 316 includes classes and interfaces used to bind
12 properties of elements to data sources, data source classes, and data-specific
13 implementations of data collections and views. These classes and interfaces are
14 also used to handle exceptions in data entry and allow runtime creation of a user
15 interface based on information in various data sources. Data can be displayed in
16 textual form or can be utilized to change the formatting of the display, such as
17 displaying dollar amounts in red if they are negative. Example classes include a
18 “Bind” class that represents a binding declaration object that manages binding
19 between a dynamic property user interface and source data, and
20 “XmlDataSource” class that serves as a data source for data binding to
21 content nodes.

22 Object-oriented applications typically represent data by classes that
23 both the value of a piece of data and the operations that can be performed on that
24 data. The term “data item” refers to one such object. Applications can handle
25



59

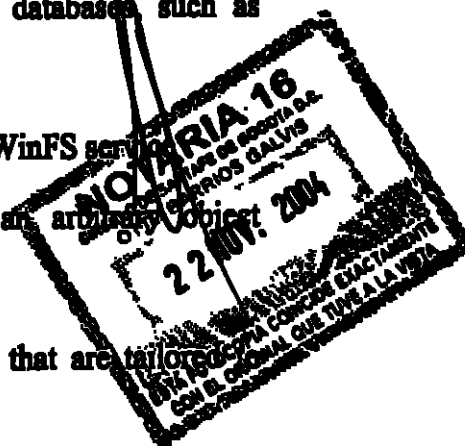
562

1 individual data items or collections of data items. They may use data items in
2 three ways: (a) converting data from external sources such as file systems, remote
3 servers, databases, etc. into the corresponding in-memory data items, and
4 converting modified data items back into the form expected by these sources; (b)
5 operating on the data items using a combination of data-centric and application-
6 centric logic; (c) presenting the data embodied by the data items to the user
7 through a user interface. Data namespace 316 provides support for the first and
8 third of these tasks.

9 The first task, obtaining data from external sources, is supported by "data
10 source" objects. A data source object is typically defined as a page-wide or
11 application-wide resource, and serves as the gateway to the data. Data sources
12 implement an IDataSource interface, which defines a standard mechanism by
13 which the classes in the data namespace get access to the data. A particular data
14 source object implements logic for retrieving the actual data by using mechanisms
15 appropriate for the particular source. In one embodiment, the data namespace
16 includes four data source classes:

- 17 1. XmlDataSource, for retrieving data represented as XML
- 18 2. SqlDataSource, for retrieving data from SQL databases such as
19 Microsoft SQL Server
- 20 3. WinFSDDataSource, for retrieving data from the WinFS service
- 21 4. ObjectDataSource, for retrieving data from an arbitrary object
22 defined by the application

23 Applications can also define their own data source classes that are tailored
24 special-purpose sources.

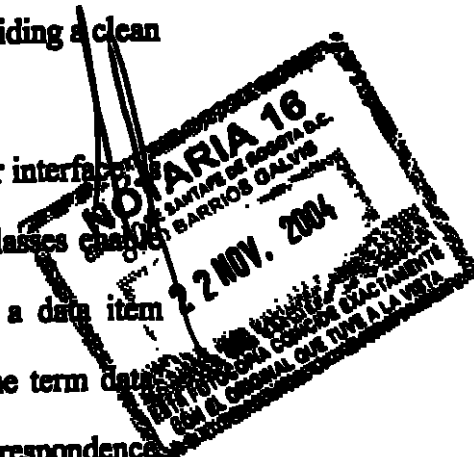


563

1 A data source class is responsible for retrieving data from an external
2 source and converting it into one or more data items suitable for use by the
3 binding classes. If a collection of data items is needed, the application can use any
4 of the standard collection classes from the .Net Framework such as Array,
5 ArrayList, Hashtable, etc., any of the data-centric collection classes from the
6 System.Data namespace such as Dataset, or a data-centric collection class from the
7 data namespace such as ArrayListDataCollection. The latter classes support
8 change notifications; i.e., when the application changes the collection by adding an
9 item, removing an item, sorting the collection, etc., the collection sends a
10 notification. The binding classes listen for these notifications and automatically
11 update the user interface to reflect the change.

12 Once the data has been converted to in-memory data items, the application
13 can perform computations using the items and can modify the items as a result of
14 the computations. These actions are performed using a combination of data-
15 centric operations (defined by the data item classes) and application-centric
16 operations (defined by the application itself). The actions may be initiated by the
17 application automatically, or in response to an action of the user. Special support
18 or cooperation from the data namespace is not necessary, thereby providing a clean
19 separation of logic and presentation within the application.

20 The third data-related task, presenting the data through the user interface,
21 supported by the "binding" classes of the data namespace. These classes enable
22 the application to describe the correspondence (binding) between a data item
23 property (the source) and a user interface property (the target). The term data
24 binding (or simply binding) refers to the establishment of such a correspondence.
25 For example, an application may choose to data-bind the Text property of a



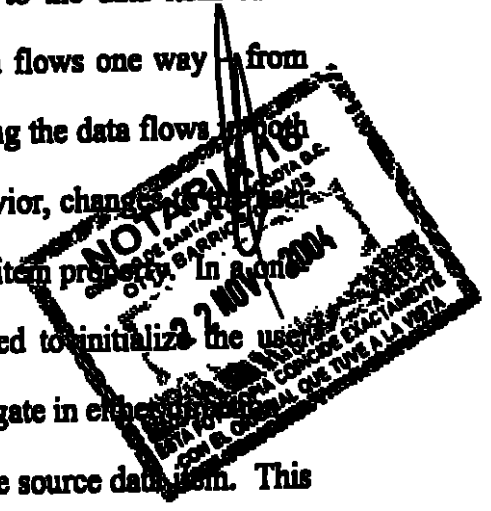
61

564

1 Textbox control to the CustomerName property of a data item. Having done so,
2 the control will automatically display the customer's name, updating the display
3 whenever the application changes the data item, and updating the data item
4 whenever the user types a new name into the control.

5 This kind of correspondence is described using the Bind class, and
6 implemented using the Binding class. Any number of UI properties can share the
7 same description (Bind), but each property has its own unique Binding that holds
8 the state for that particular instance. The description includes the following
9 information about the desired correspondence:

- 10 • Path – the name of the data item property to use as the source of the
11 binding. This can be a simple property name, or a more complicated
12 expression involving sub-objects and indexers (e.g., when the source
13 property has a value of complex type) such as
14 "ShippingAddress.Line[2]". When the data source is XML, the path
15 is an XPath expression.
- 16 • BindType – whether the correspondence is one-way, two-way, or
17 one-time. In a one-way binding, changes to the data item cause
18 updates to the user interface property; data flows one way – from
19 the source to the target. In a two-way binding the data flows in both
20 directions; in addition to the one-way behavior, changes to the user
21 interface property cause updates to the data item property. In a one-time
22 binding, the data item property is used to initialize the user interface
23 property, but changes do not propagate in either direction.
- 24 Source – a description of where to obtain the source data item. This
25 can be from a data source object, from some other user interface



62

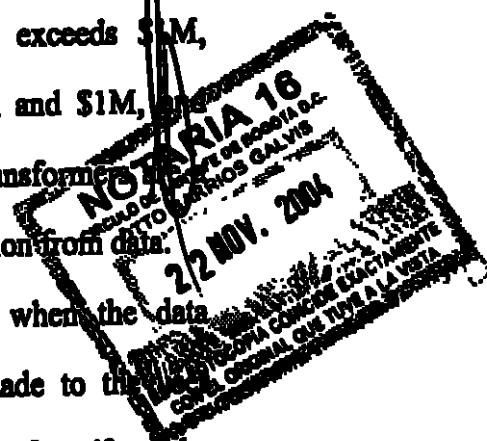
565

1 element, or from the value of the target element's DataContext
2 property.

3 UpdateType - when to update the source property in a two-way
4 binding: one of Immediate, OnLostFocus, or Explicit. Immediate
5 updates happen as soon as the user interface property changes.
6 OnLostFocus updates are delayed until the target element loses
7 keyboard focus - this is appropriate for a TextBox control, to avoid
8 the expense of updating after every keystroke. Explicit updates
9 happen when the application explicitly calls for them.

- 10 • Transformer - an object that implements the IDataTransformer
11 interface. This gives the application a way to modify the data item
12 value before using it in the user interface. The modification can be a
13 simple type conversion (e.g., in a binding of the Background
14 property to the BalanceOwed data property, the application can
15 convert a negative balance to a red background and a positive
16 balance to green), or an application-specific conversion (e.g., in a
17 binding of the Text property to the NetWorth data property, the
18 application can display "Rich" if the NetWorth exceeds \$1M,
19 "Bourgeois" if the NetWorth lies between \$100K and \$1M, and
20 "Poor" if the NetWorth is less than \$100K). Transformer is a
21 simple yet powerful tool that help separate presentation from data.

22 All bindings except one-time rely on getting notified when the data
23 property changes, so that the corresponding change can be made to the
24 interface. The binding classes recognize the IPropertyChange interface (from the
25



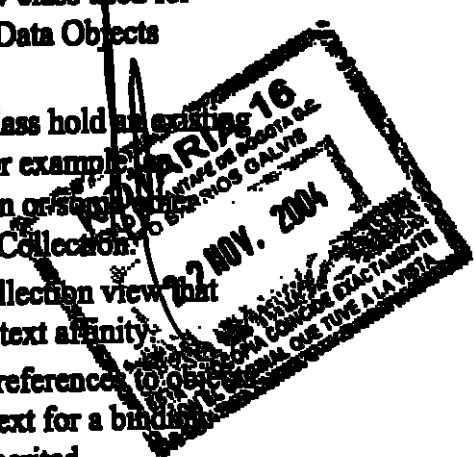
566

System.ComponentModel namespace) as one way of implementing the required notifications.

The following tables list the members exposed by the System.Windows.Data namespace.

Classes

ArrayListCollectionView	Encapsulates the collection view support for the ArrayListDataCollection collection class. This class cannot be inherited.
ArrayListDataCollection	Provides a built-in implementation of an array-list data collection with an underlying collection-view interface. It also implements ICollectionChange to provide notification when items are added, items are removed, or the whole collection is refreshed.
Bind	Represents a bind declaration object, used to manage bindings between a dynamic property user interface (UI) and source data.
Binding	Provides access to the single run-time instance of a binding. This class cannot be inherited.
BindingListCollectionView	A collection view class used for Microsoft® ActiveX® Data Objects (ADO) data views.
CollectionContainer	Objects of this class hold an existing collection structure—for example, ArrayListDataCollection or DataSet inside the ItemCollection.
ContextAffinityCollectionView	Implements a collection view that includes checks for context affinity.
DataContextObjectRef	Supports object references to objects being used as data context for a binding. This class cannot be inherited.
DataSourceObjectRef	Supports object references to data

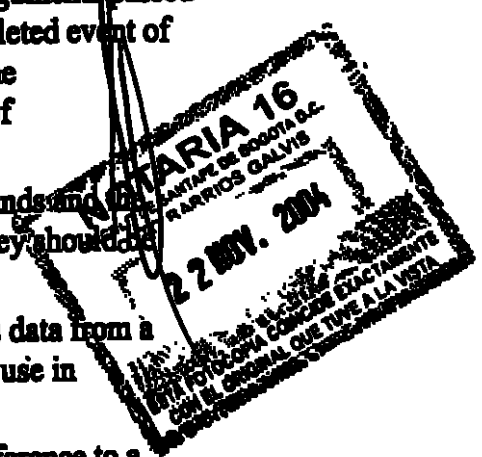


64

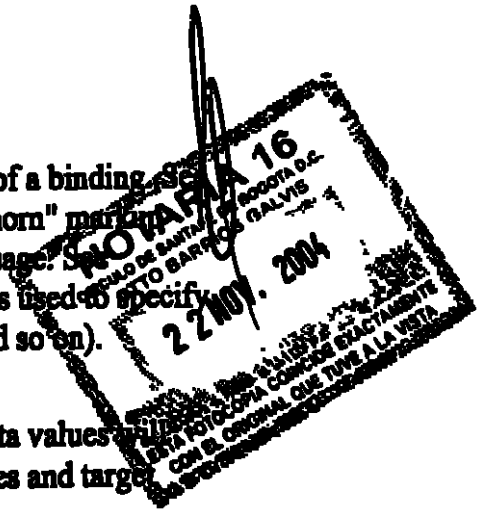
567

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

	sources. This class cannot be inherited.
DataTransferEventArgs	Encapsulates arguments for data transfer events. The events are routed events that are handled specifically by a designated handler based on the DataTransferEventHandler delegate.
ElementObjectRef	Represents an object reference to an element, with the object being specified by its element ID. This class cannot be inherited.
ExplicitObjectRef	Represents an explicit object reference to an element. This class cannot be inherited.
ListCollectionView	Implements a collection view for collections based on IList.
ObjectDataSource	Serves as a data source for data binding. Bindable data items can be specified as common language runtime types.
ObjectRef	The abstract class that is used as the parent class of ElementObjectRef, ExplicitObjectRef, and TypeObjectRef.
ParameterCollection	Objects of this class hold the collection of named parameters (with their corresponding values) for an SqlDataSource.
QueryCommand	This class represents a single select statement to be submitted to the database.
RefreshCompletedEventArgs	Encapsulates the arguments passed either in the RefreshCompleted event of ObjectDataSource, or in the RefreshCompleted event of XmlDataSource.
SqlCommandList	A list of sql commands and the names of the tables that they should be used to fill.
SqlDataSource	SqlDataSource gets data from a Microsoft SQL Server for use in databinding.
TransformerSource	Allows resource reference to a transformer class that is defined as code-



		behind in the current application.
1	TypeObjectRef	Supports object reference by type. This class cannot be inherited.
2		
3	WinFSDataSource	The WinFSDataSource facilitates databinding of data stored in WinFS with Avalon applications
4		
5	XmlDataNamespaceManager	XmlDataNamespaceManager Class Used to declare namespaces to be used in Xml data binding XPath queries
6		
7	XmlDataSource	Serves as a data source for data binding to Extensible Markup Language (XML) content nodes.
8		
9	XmlNamespace	Declares an individual namespace within an XML data source.
10		
11	Interfaces	
12		
13	IContains	Used to create classes that declare filtering criteria for collection views.
14	IDataSource	Supports creation of data source objects. Data source objects are used for common representation of data for data binding.
15		
16	IDataTransformer	Provides methods that enable client-side transformation of bound data.
17		
18	Enumerations	
19		
20	BindFlags	Describes special properties of a binding. Using Bind Declarations for "Longhorn" markup language (code-named "XAML") usage. See BindType for the enumeration that is used to specify binding type (one-way, two-way and so on).
21		
22	BindStatus	Status of a Binding.
23		
24	BindType	Describes how changes in data values transfer to and from source properties and target properties of a binding.
25	SqlDataSourceMode	The enumeration of the possible modes that



569

1 **SqlDataSource** can have. The mode determines what
2 sort of data is returned when the application retrieves
3 the value from the **Data** property.

4 **UpdateType** Specifies when updates to the data source
5 (target-to-source data transfer) should occur in a
6 binding. Setting these values will only be relevant if a
7 binding's **BindType** is set to **TwoWay** (or left as the
8 default).

9 **Delegates**

10 **DataChangedEventHandler** Represents the method that handles
11 the **DataChanged** event raised by data
12 sources that implement **IDataSource**.

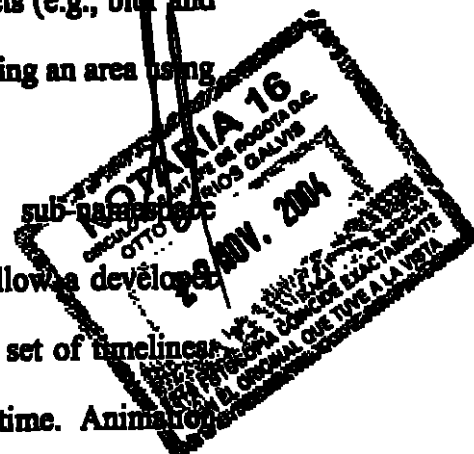
13 **DataTransferEventHandler** Represents the method that handles
14 a data transfer event raised by **Binding**.

15 **RefreshCompletedEventHandler** Represents the method that handles
16 the **ObjectDataSource.RefreshCompleted**
17 and **XmlDataSource.RefreshCompleted**
18 events.

19 **Media Namespace**

20 A media namespace 318 provides various media classes. Application
21 developers as well as component developers may use these classes to develop
22 various presentation functionality. Example classes in media namespace 318
23 include an **"ImageEffect"** class that permits certain imaging effects (e.g., **bhr** and
24 grayscale), and a **"Brush"** class that provides a mechanism for filling an area using
25 solid colors, gradients, images, video, and the like.

26 The media namespace 318 includes a sub-namespace
27 **System.Windows.Media.Animation** that includes services that allow a developer
28 to animate properties and coordinate a set of animations with a set of timelines.
29 An animation is an object that changes a value over a period of time. Animation
30 effects include moving an object on the display, and changing the size, shape, or



68

560
510

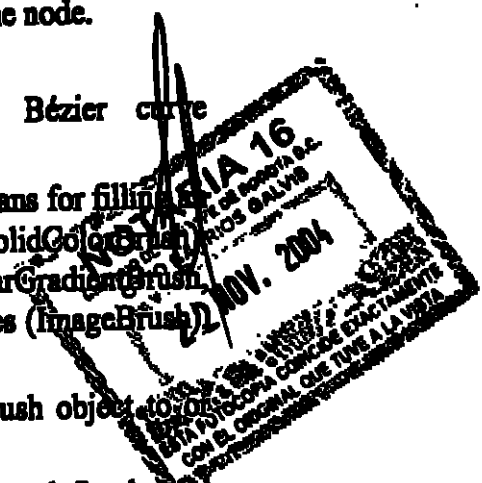
color of an object. Multiple animation classes are provided to implement various animation effects. Effects can be achieved by associating an animation with an element's property value. For example, to create a rectangle that fades in and out of view, one or more animations are associated with the opacity property of the rectangle.

The media namespace 318 also includes a sub-namespace System.Windows.Media.TextFormatting that provides various text services. For example, a "TextFormatter" text engine provides services for breaking text lines and formatting text presented on a display. "TextFormatter" is capable of handling different text character formats and paragraph styles as well as handling international text layout.

The following tables list example members exposed by the System.Windows.Media namespace.

Classes

ArcSegment	Represents an elliptical arc between two points.
AudioData	Enables playing of audio files according to the state of a time node.
AudioDataConverter	AudioDataConverter
BezierSegment	Represents a cubic Bézier curve drawn between two points.
Brush	Provides a generic means for filling an area using solid colors (SolidColorBrush), gradients (LinearGradientBrush, RadialGradientBrush), images (ImageBrush), video, and more.
BrushConverter	Used to convert a Brush object to or from another object type.
Brushes	Implements a set of predefined solid colors.

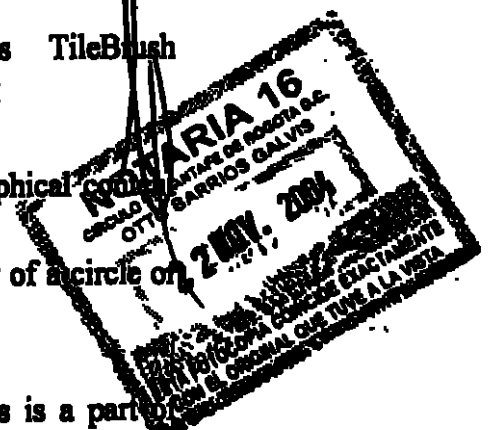


68

5+1

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

CloseSegment	Represents a line that connects the last point of a PathFigure object with its starting point.
CodecFilter	Filter for enumerating codecs. Only those codecs that match the properties will be enumerated.
CodecInfo	Information about a specific codec and a factory for creating the codec. This is returned from the codec enumerator.
ColorCollection	
ColorCollectionConverter	ColorCollectionConverter - Converter class for converting instances of other types to and from ColorCollection instances.
ColorContext	
ColorConverter	Used to convert a Color object to or from another object type.
Colors	Implements a set of predefined colors.
ContainerVisual	Manages a collection of Visual objects.
DashArrays	DashArrays - The DashArrays class is static, and contains properties for well known dash styles.
DoubleCollection	
DoubleCollectionConverter	DoubleCollectionConverter - Converter class for converting instances of other types to and from DoubleCollection instances.
Drawing	A Drawing is a list of 2d drawing primitives.
DrawingBrush	DrawingBrush - This TileBrush defines its content as a Drawing
DrawingContext	Drawing context.
DrawingVisual	Visual that contains graphical content to be drawn.
EllipseGeometry	Represents the geometry of a circle or ellipse.
FontFamily	Font family
FormattedText	The FormattedText class is a part of Avalon MIL easy text API, which is targeted at programmers needing to add some simple



67

542

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Geometry

text to a MIL visual.

An abstract class that provides base functionality for all geometry classes, such as EllipseGeometry, RectangleGeometry, and PathGeometry. The Geometry class of objects can be used for clipping, hit-testing, and rendering 2-D graphic data.

GeometryCollection

Represents a collection of Geometry objects.

GetPageEventArgs

class GetPageEventArgs

GlyphRun

Glyph run class

GlyphTypeface

Physical font face corresponds to a font file on the disk

GradientBrush

An abstract class that describes a gradient fill. Classes that derive from GradientBrush describe different ways of interpreting gradient stops.

GradientStop

Describes the location and color of a transition point in a gradient.

GradientStopCollection

Represents a collection of GradientStop gradient stops.

HitTestParameters

This is the base class for packing together parameters for a hit test pass.

HitTestResult

This base returns the visual that was hit during a hit test pass.

HwndInterop

HwndInterop

HwndVisual

HyphenationCandidate

Describes one Hyphenation candidate.

ICCProfile

ImageBrush

Fills an area with an image. This class may be used to specify images as the background of other objects.

ImageCodecCollection

The collection of codecs (actually CodecInfos) on the system.

ImageCodecEnumerator

The enumerator for Image frames.

ImageColorTransform

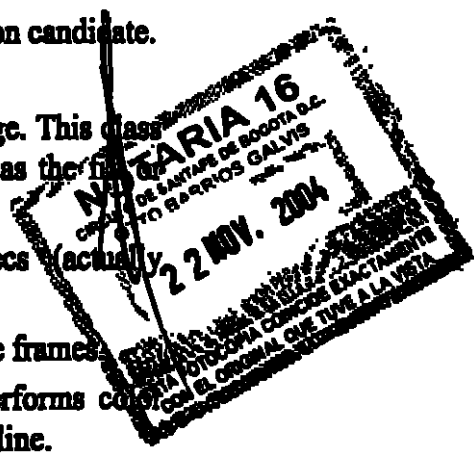
ImageColorTransform Performs color management on an imaging pipeline.

ImageData

Contains an image and related data.

ImageDataBuilder

This object is used to build an



70

573

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ImageDecoder

ImageData object.

ImageDecoder is a container for image frames. Each image frame is an ImageSource. Unlike ImageSource, ImageDecoder is NOT an immutable object and can be re-initialized to a different image stream. However, any ImageSources (frames) that it returns should be immutable.

ImageDecoderBmp

The built-in Microsoft Bmp (Bitmap) Decoder.

ImageDecoderGif

The built-in Microsoft GIF Decoder.

ImageDecoderIcon

The built-in Microsoft Icon Decoder.

ImageDecoderInternal

For internal use only:

ImageDecoderJpeg

The built-in Microsoft Jpeg Decoder.

ImageDecoderPng

The built-in Microsoft PNG Decoder.

ImageDecoderTiff

The built-in Microsoft Tiff Decoder.

ImageEffect

The ImageEffect class is the base class for all imaging effects (blur, grayscale, etc) It's possible for an effect to not have any inputs but an effect should have at least one output. The default implementations of things assume this. If a derived effect is going to play with Output/Outputs be sure that at least one is there.

ImageEffectBlur

Gaussian blur effect. It is a single input, single output effect. Warning: If the effect is being scaled (i.e. Input.ScaleX or Input.ScaleY isn't 1) and Expand is true, then it's possible for the output dimensions to be larger or smaller than PixelWidth and PixelHeight. Adjust the pixel buffer feed copy to avoid problems.

ImageEffectFlipRotate

This effect can flip an image in and rotate by multiples of 90 deg

ImageEffectGammaCorrect

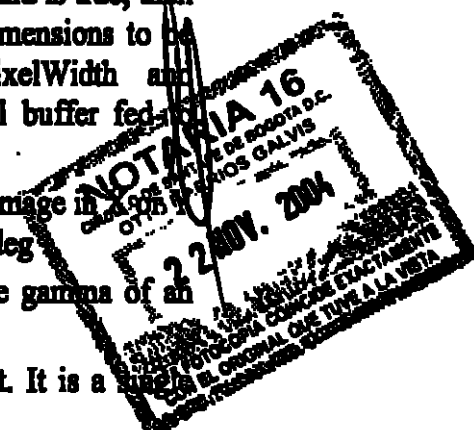
This effect changes the gamma of an image

ImageEffectGlow

Performs a glow effect. It is a single input, single output effect.

ImageEffectGrayscale

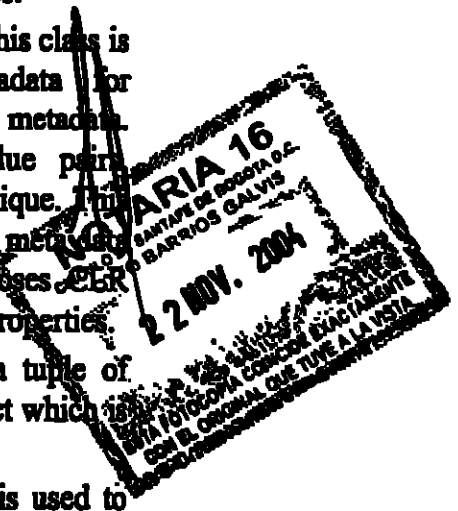
Converts an image to grayscale. It is a single input, single output effect.



571

574

1	ImageEffectNegate	Negates an image. It is a single input, single output effect.
2	ImageEffectSharpen	Unsharp mask. It is a single input, single output effect.
3	ImageEffectSource	ImageEffectSource class implementation
4	ImageEffectSourceCollection	The collection of image effect outputs
5	ImageEffectTint	Tint constructor. It is a single input, single output effect.
6		
7		ImageEncoder collects a set of frames (ImageSource's) with their associated thumbnails and metadata and saves them to a specified stream. In addition to frame-specific thumbnails and metadata, there can also be an image-wide (global) thumbnail and metadata, if the codec supports it.
8	ImageEncoder	
9		
10	ImageEncoderBmp	Built-in Encoder for Bmp files.
11	ImageEncoderGif	Built-in Encoder for Gif files.
12		ImageEncoderInternal collects a set of frames (ImageSource's) with their associated thumbnails and metadata and saves them to a specified stream. In addition to frame-specific thumbnails and metadata, there can also be an image-wide (global) thumbnail and metadata, if the codec supports it.
13	ImageEncoderInternal	
14		
15		
16	ImageEncoderJpeg	Built-in Encoder for Jpeg files.
17	ImageEncoderPng	Built-in Encoder for Png files.
18	ImageEncoderTiff	Built-in Encoder for Tiff files.
19		ImageExchangeMetaData This class is used to access and set metadata for ImageFiles which have Exif style metadata. MetaData is stored as Key/Value pairs where Keys are not necessarily unique. This class provides generic access to all metadata within an image, as well as exposes CLR properties for certain well-known properties.
20	ImageExchangeMetaData	
21		
22		
23	ImageExchangeProperty	ImageExchangeProperty - a tuple of an ImageExchangeID and the object which is the value of that property
24		
25	ImageMetaData	ImageMetaData This class is used to

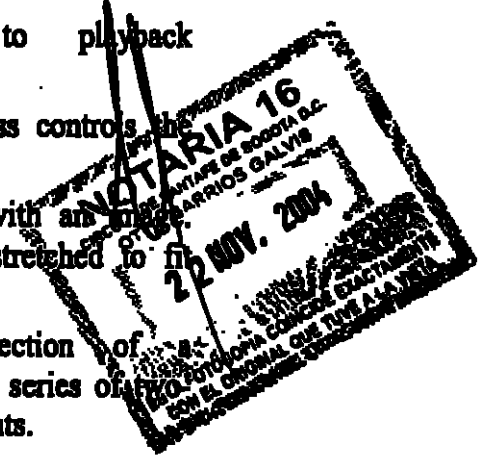


72

5x5

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

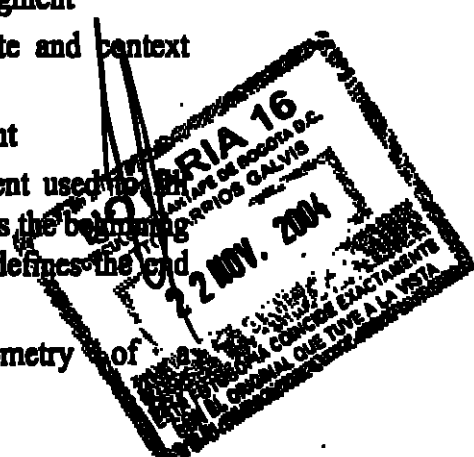
	access and set metadata for Images. This class also exposes a CodecMetaData property which exposes a codec-specific means of accessing the metadata for this image.
ImagePalette	ImagePalette class
ImageSizeOptions	Sizing options for an image. The resulting image will be scaled based on these options.
ImageSource	Defines the methods, properties, and events for the imaging pipeline, including decoders and effects.
ImageSourceCollection	The collection of codecs (actually ImageSource's) on the system.
ImageSourceConverter	ImageSourceConverter
IntegerCollection	
IntegerCollectionConverter	IntegerCollectionConverter Converter class for converting instances of other types to and from IntegerCollection instances.
LinearGradientBrush	Defines a linear gradient used to fill an area.
LineGeometry	Represents the geometry of a line.
LineSegment	Represents a line between two points. Unlike LineGeometry objects, LineSegment should be contained within a PathFigure.
MatrixTransform	Creates an arbitrary affine matrix transformation used to manipulate objects or coordinate systems in a two-dimensional plane.
MediaData	MediaData. Use to playback Audio/Video content.
MediaSystem	The MediaSystem class controls the media layer.
NineGridBrush	Fills an entire area with an image. Portions of the image are stretched to fit within defined margins.
PathFigure	Represents a sub-section of geometry, a single connected series of dimensional geometric segments.
PathFigureCollection	



23

546

1	PathFigureConverter	PathFigureConverter
2	PathGeometry	Represents a complex shape that may be composed of arcs, curves, ellipses, lines, and rectangles.
3	PathGeometryConverter	PathGeometryConverter
4		An abstract class that represents a segment of a PathFigure object. Classes that derive from PathSegment, such as ArcSegment, BezierSegment, and LineSegment, represent specific types of geometric segments.
5	PathSegment	
6		
7	PathSegmentCollection	Represents a list of PathSegment objects.
8	PathSegmentConverter	PathSegmentConverter
9	Pen	Describes how a shape is outlined.
10	PixelFormats	PixelFormats - The collection of supported Pixel Formats
11	PointCollection	
12	PointCollectionConverter	PointCollectionConverter - Converter class for converting instances of other types to and from PointCollection instances.
13		
14	PointHitTestParameters	This is the class for specifying parameters hit testing with a point.
15	PointHitTestResult	This class returns the point and visual hit during a hit test pass.
16	PolyBezierSegment	PolyBezierSegment
17	PolyLineSegment	PolyLineSegment
18	PolyQuadraticBezierSegment	PolyQuadraticBezierSegment
19	PrintContext	PrintContext holds state and context for a printer interaction
20	QuadraticBezierSegment	QuadraticBezierSegment
21	RadialGradientBrush	Defines a radial gradient used for an object. A focal point defines the beginning of the gradient, and a circle defines the end point of the gradient.
22		
23	RectangleGeometry	Represents the geometry of rectangle.
24	RetainedVisual	RetainedVisual
25	RotateTransform	Used to rotate an object about a



74

5-17

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ScaleTransform

specified point in the two-dimensional x-y plane.

Scales an object in the two-dimensional x-y plane, starting from a defined center point. Scale factors are defined in x- and y-directions from this center point.

SkewTransform

Represents a two-dimensional skew.

SolidColorBrush

Represents a solid, uniform fill.

StartSegment

StartSegment

SubLineCollection

collection of subline. Subline can be object of one of these types GlyphRun LineOver Inline object

TileBrush

Abstract class that describes a way to fill a region with one or more "tiles." Derived classes define the different types of tiles that can be used; for example, the ImageBrush enables you to fill an area with an image.

Transform

An abstract class that you use as the parent class of all types of transformations in a two-dimensional plane, including rotation (RotateTransform), scale (ScaleTransform), skew (SkewTransform), and translation (TranslateTransform). This class hierarchy differs from the Matrix structure both because it is a class and because it supports animation and enumeration semantics.

TransformCollection

Used to create and manipulate a list of Transform objects.

TransformConverter

Used to convert a Transform object to or from another object type.

TranslateTransform

Translates an object in the two-dimensional x-y plane.

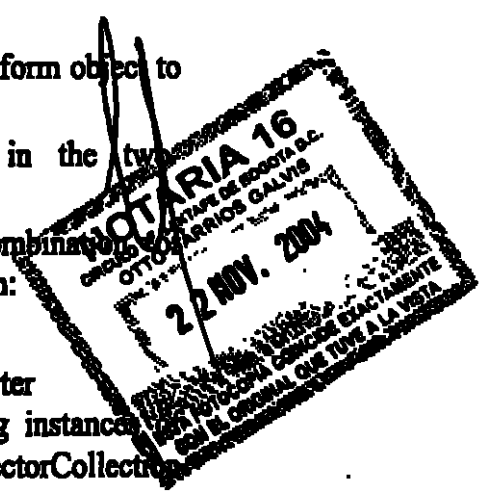
Typeface

A Typeface is a combination of family, weight, style and stretch:

VectorCollection

VectorCollectionConverter

VectorCollectionConverter Converter class for converting instances of other types to and from VectorCollection instances.

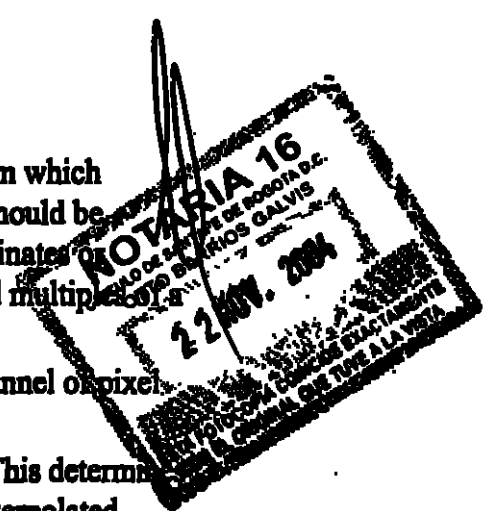


75

578

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

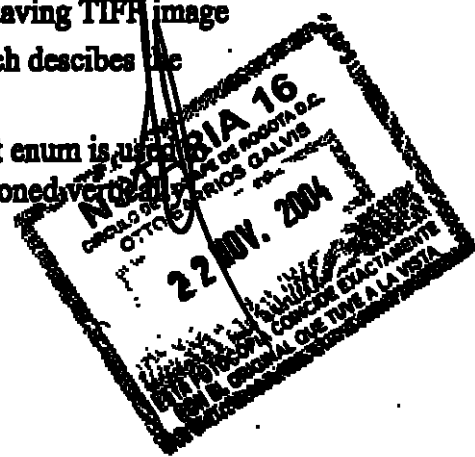
VideoData	Enables playing of video files according to the state of a time node.
VideoDataConverter	VideoDataConverter
Visual	Base class for all Visual types. It provides services and properties common to all Visuals, including hit-testing, coordinate transformation, and bounding box calculations.
VisualCollection	An ordered collection of Visual objects.
VisualManager	Renders a tree of Visual objects to a rendering target, typically a window.
Interfaces	
IHyphenate	IHyphenate is the interface for Hyphenation Service Provider
IRetainedRender	If this interface is implemented on a class that is derived from a RetainedVisual, the RetainedVisual operations in validation mode, i.e. the graphics subsystem will call OnRender in a lazy fashion. (e.g. if the Visual appears for the first time on the screen). Note that OnRender can be called by the system anytime.
IVisual	This interface defines the common methods and services available from a Visual object.
Enumerations	
BrushMappingMode	BrushMappingMode - Enum which describes whether certain values should be considered as absolute local coordinates or whether they should be considered multiple of bounding box's size.
ChannelDescription	Describes order of each channel of pixel data
ColorInterpolationMode	ColorInterpolationMode - This determines how the colors in a gradient are interpolated.



76

549

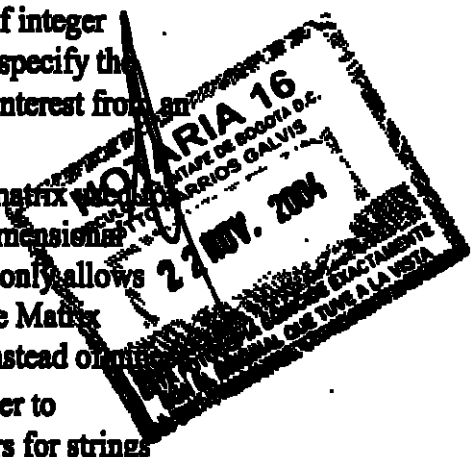
1	CombineMode	Specifies the method used to combine two geometric areas.
2	FillRule	
3	GradientSpreadMethod	Specifies how the gradient should be drawn outside of the specified gradient vector or space.
4	HitTestFilterBehavior	Behavior for filtering visuals while hit testing
5	HitTestResultBehavior	Enum controls behavior when a positive hit occurs during hit testing.
6	HorizontalAlignment	The HorizontalAlignment enum is used to describe how content is positioned horizontally within a container.
7	HyphenationRule	Supported Hyphenation Rules.
8	ImagePaletteType	Pre-defined palette types
9	MediaState	Holds the current state of the Media
10	PenDashCap	PenDashCap - Enum which describes the drawing of the ends of a dash within a dashed line.
11	PenLineCap	Describes the shape at the end of a line or segment.
12	PenLineJoin	PenLineJoin - Enum which describes the drawing of the corners on the line.
13	Rotation	The rotation to be applied; only multiples of 90 degrees is supported.
14	StandardColorSpace	
15	Stretch	Stretch - Enum which describes how a source rect should be stretched to fit a destination rect.
16	StyleSimulations	Font style simulation
17	TiffCompressOptions	Compress options for saving TIFF image
18	TileMode	TileMode - Enum which describes the drawing of the ends of a line.
19	VerticalAlignment	The VerticalAlignment enum is used to describe how content is positioned vertically within a container.
20	Structures	
21		
22		
23		
24		
25		



27

570
580

1	CharacterIndexer	This class is a helper to implement named indexers for characters.
2		
3	Color	Represents colors in terms of alpha, red, green, and blue channels.
4		
5	GlyphIndexer	This class is a helper to implement named indexers for glyph metrics.
6		
7	ImageExchangeID	ImageExchangeID - This class is the type which can be used as the key for a property in an ImageMetaData instance. This can be either an integer or a string.
8		
9	ImageExchangeMetaDataEnumerator	ImageExchangeMetaDataEnumerator The enumerator for ImageExchangeMetaData . Contains IEnumerator interface as well as strongly typed versions of the APIs
10	ator	
11		
12	ImageFrameEnumerator	The enumerator for Image frames.
13		
14	ImageMetaDataRational	An ImageMetaDataRational class is represented as a signed numerator and a signed denominator. The effective value of a rational is the numerator / demoninator
15		
16		
17	ImageMetaDataUnsignedRational	A rational class is represented as an unsigned numerator and an unsigned denominator. The effective value of a rational is the numerator / demoninator
18		
19	ImagePaletteColor	ImagePaletteColor structure
20		
21	IntegerRect	A rect composed of integer values. Typically used to specify the source rect (in pixels) of interest from an image.
22		
23	Matrix	Represents a 3x3 matrix of transformations in two-dimensional space. Because "Avalon" only allows affine transformations, the Matrix structure has six entries instead of nine.
24		
25	NamedStringIndexer	This class is a helper to implement named indexers for strings



XB

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

PixelFormat localized in multiple cultures.
Pixel Format Definition for
images and pixel-based surfaces

Delegates

GetPageEventHandler delegate GetPageEventHandler
HitTestFilterDelegate Delegate for hit tester to control whether to
test against children of visual.
HitTestResultDelegate Delegate for hit tester to control returning of
hit information on visual.

The following tables list example members exposed by the
System.Windows.Media.Animation namespace.

Classes

Animatable Any class that doesn't derive from
DependencyObject but which has
properties that can be animated should
derive from this class.

AnimationCollection This abstract class provides base
functionality for animation collections,
such as ColorAnimationCollection,
DoubleAnimationCollection, and
SizeAnimationCollection.

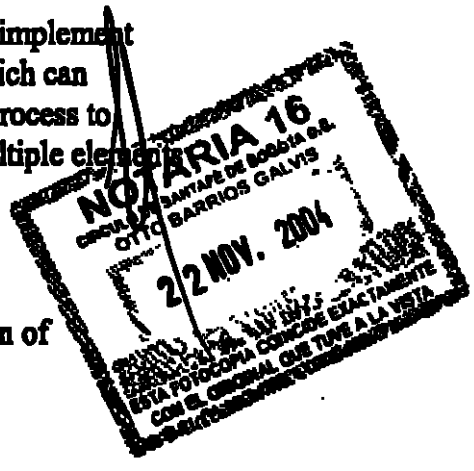
AnimationEffect Override this class to implement
element level animations which can
participate in the rendering process to
instantiate animations on multiple elements
at rendering time.

AnimationEffectCollection Holds a collection of
AnimationEffects.

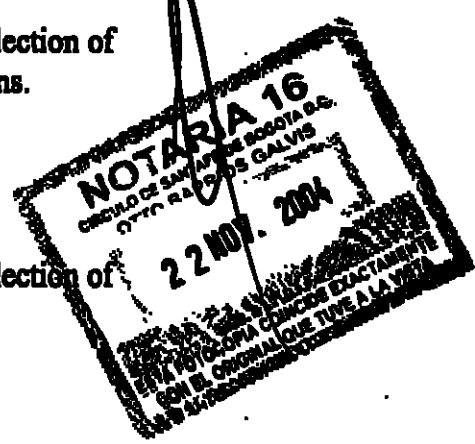
BoolAnimationCollection Represents a collection of
BoolModifier animations.

BoolModifier

BoolTimedModifier

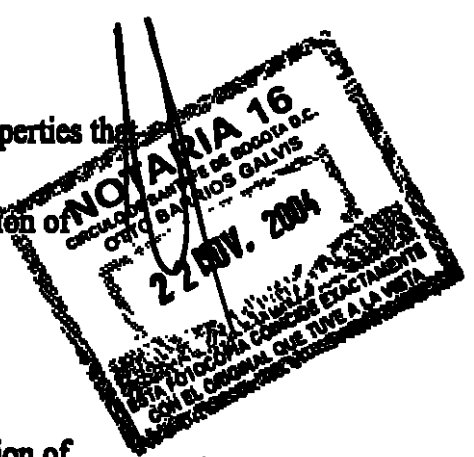


1	ByteAnimationCollection	Represents a collection of BoolModifier animations.
2	ByteModifier	
3	ByteTimedModifier	
4	CharAnimationCollection	Represents a collection of CharModifier animations.
5	CharModifier	
6	CharTimedModifier	
7	ColorAnimation	Animates a color value of a property.
8	ColorAnimationCollection	Represents a collection of ColorModifier animations.
9	ColorKeyFrameCollection	
10	ColorModifier	
11	ColorTimedModifier	
12	DecimalAnimationCollection	Represents a collection of DecimalModifier animations.
13	DecimalModifier	
14	DecimalTimedModifier	
15	DoubleAnimation	Used to animate properties that accept a Double value.
16	DoubleAnimationCollection	Represents a collection of DoubleModifier animations.
17	DoubleKeyFrameCollection	
18	DoubleModifier	
19	DoubleTimedModifier	
20	FloatAnimation	Used to animate properties that accept a Single value.
21	FloatAnimationCollection	Represents a collection of FloatModifier animations.
22	FloatKeyFrameCollection	
23	FloatModifier	
24	FloatTimedModifier	
25	IntAnimationCollection	Represents a collection of IntModifier animations.
	IntModifier	
	IntTimedModifier	
	LengthAnimation	Used to animate properties that



583

1		accept a Length value.
2	LengthAnimationCollection	Represents a collection of LengthModifier animations.
3	LengthKeyFrameCollection	
4	LengthModifier	
5	LengthTimedModifier	
6	LongAnimationCollection	Represents a collection of LongModifier animations.
7	LongModifier	
8	LongTimedModifier	
9	MatrixAnimationCollection	Represents a collection of MatrixModifier animations.
10	MatrixModifier	
11	MatrixTimedModifier	
12	Modifier	
13	ObjectAnimationCollection	Represents a collection of ObjectModifier animations.
14	ObjectModifier	
15	ObjectTimedModifier	
16	PathAnimation	This animation can be used inside of a MatrixAnimationCollection to move a visual object along a path.
17	PointAnimation	Used to animate properties that accept Point values.
18	PointAnimationCollection	Represents a collection of PointModifier animations.
19	PointKeyFrameCollection	
20	PointModifier	
21	PointTimedModifier	
22	RectAnimation	Used to animate properties that accept a Rect value.
23	RectAnimationCollection	Represents a collection of RectModifier animations.
24	RectKeyFrameCollection	
25	RectModifier	
	RectTimedModifier	
	ShortAnimationCollection	Represents a collection of ShortModifier animations.



81

584

ShortModifier

ShortTimedModifier

SizeAnimation

Defines an animation based on the Size of an object. By providing Size information, an object can appear to shrink or enlarge over a period of time.

SizeAnimationCollection

Represents a collection of SizeModifier animations.

SizeKeyFrameCollection

SizeModifier

SizeTimedModifier

StringAnimationCollection

Represents a collection of StringModifier animations.

StringModifier

StringTimedModifier

Timeline

Maintains run-time timing state for timed objects.

TimelineBuilder

An object that can be used to create Timeline objects.

TimeManager

The object that controls an entire timing tree.

TimeSyncValueTypeConverter

An object that performs type conversions involving TimeSyncValue values.

TimeTypeConverter

An object that performs type conversions involving Time values.

VectorAnimation

Used to animate properties that accept a Vector value.

VectorAnimationCollection

Represents a collection of VectorModifier animations.

VectorKeyFrameCollection

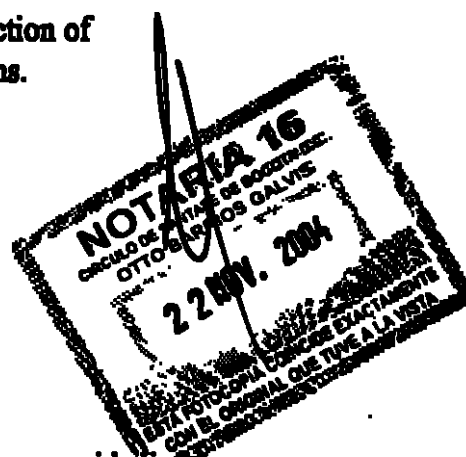
VectorModifier

VectorTimedModifier

Interfaces

IClock

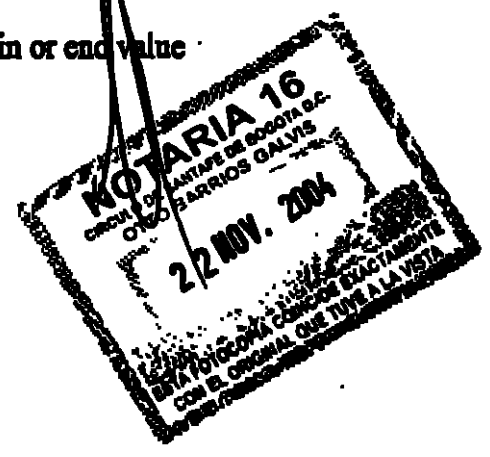
Represents an object that can provide linear



82

585

1		time values.
2	IModifier	Defines the basic behavior of a modifier object. A modifier is an object that takes an object, called the <i>base value</i> , of a certain type and returns another object of the same type as its output.
3		
4	ITimingControl	Defines the behavior of timelines and timed objects.
5	ITimingControlBuilder	Represents an object that can build a timeline template.
6		
7		
8	Enumerations	
9	AnimationType	Describes the behavior of an animation.
10	CloneType	The types of clones that CloneCore may request.
11	InterpolationMethod	Describes how an animation calculates its output values.
12	KeyTimeType	The different types of KeyTimes
13	TimeEndSync	Values for the endSync attribute, which specifies how a container calculates its simple duration based on the children's durations.
14		
15	TimeFill	Specifies how the timeline behaves after it is no longer active.
16	TimeRestart	Values for the Timeline.Restart attribute.
17	TimeSeekOrigin	Indicates a timeline position; used to specify the behavior of the ITimingControl interface's Seek method by defining the position to which the offset is applied.
18		
19	TimeSyncBase	The event to synchronize a begin or end value to.
20		
21		
22	Structures	
23	ColorKeyFrame	
24	DoubleKeyFrame	
25	FloatKeyFrame	



013

586

1	KeySpline	This class is used to pass an array of key splines into the KeySplines property of an animation fragment.
2		
3	KeyTime	A KeyTime is use to specify when relative to the time of an animation that a KeyFrame takes place.
4	LengthKeyFrame	
5	PointKeyFrame	
6	RectKeyFrame	
7	SizeKeyFrame	
8	Time	A value representing time, with associated time arithmetic operations.
9	TimelineEnumerator	Enumerates items in an TimelineList collection.
10	TimeSyncValue	A value representing an absolute or relative begin or end time for a timeline.
11	VectorKeyFrame	

12 The following tables list example members exposed by the
13 System.Windows.Media.TextFormatting namespace.

14 Classes

15		
16		Provides measurement details for inline text objects. The formatting client passes this object as a parameter to the GetInlineObjectInfo method.
17	InlineObjectInfo	
18		TextFormatter is the "Avalon" text engine and provides services for formatting text and breaking text lines. TextFormatter can handle different text character formats and paragraph styles, and includes support for international text layout.
19		
20	TextFormatter	
21		
22	TextHighlightBounds	Bounds of text range
23	TextInfo	Represents information about a block of text in the client's text source character store.
24		
25	TextLine	Provides services to a line of text. Inherit from this class to implement services



84

587

		that manipulate and format a line of text. This is an abstract class.
1		
2	TextMarkerGeneratedContent	Generates line list marker output.
3		Defines the style and type of a
4	TextMarkerInfo	paragraph's list marker. The formatting client uses this class as a parameter to provide marker details to the GetTextMarkerInfo method.
5		
6	TextParagraphProperties	Represents properties that can change from one paragraph to the next, such as flow direction, alignment, or indentation.
7		
8	TextRun	Defines a sequence of characters that share a single property set. The formatting client provides TextRun details into this class when the TextFormatter passes it as a parameter to the GetTextRun method.
9		
10	TextRunBounds	Bounds of text run
11		
12	TextRunCache	Provides caching services to the TextFormatter object in order to improve performance.
13		
14	TextRunClientData	Represents client information data associated with a TextRun.
15		
16	TextRunProperties	Provides properties that can change from one TextRun to another, such as typeface or foreground brush. This is an abstract class.
17		
18	TextRunTypographyProperties	Provides typography properties for TextRun. This client set of properties generates a set of features that are processed by the OpenType layout engine.
19		
20	TextSource	Provides character data and formatting properties to the TextFormatter. All access to the text in the TextSource is achieved through the GetTextRun method, which is designed to allow the client to virtualize text in any way it chooses. This is an abstract class.
21		
22		
23		
24	TextTrimmingInfo	Provides description of text trimming characteristics. The formatting client fills trimming details into this class when the TextFormatter passes it as a parameter of
25		



85

the GetTextTrimmingInfo method.

Enumerations

TextParagraphFlags	Flags describing paragraph characteristics
TextRunCacheFlags	Kind of content in text run cache
TextRunType	Indicates the type of TextRun.

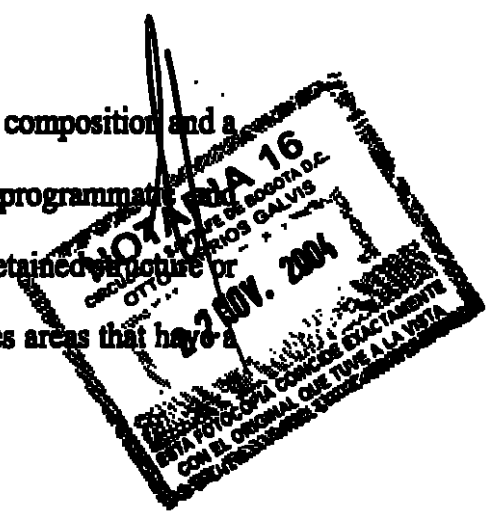
Structures

TextSourceCharacterIndex	TextSourceCharacterIndex represents a caret or character position in text.

Media Integration Layer

A Media Integration Layer (MIL) provides an API for developers or programmers to accomplish possibly complex composition effects within their applications in a straightforward manner, while leveraging the graphics processing unit in a manner that does not adversely impact normal application performance. One aspect of the MIL provides the ability to combine different media types (e.g., 2D, 3D, Video, Audio, text, imaging and so forth) and animate them together smoothly and seamlessly.

The MIL provides a graphics architecture for multi-stage composition and a programming model that allows for functional parity at the programmer scripted interfaces. An API and script allows the creation of a retained structure or scene description that is composited when rendered, yet includes areas that have a more immediate-mode nature.

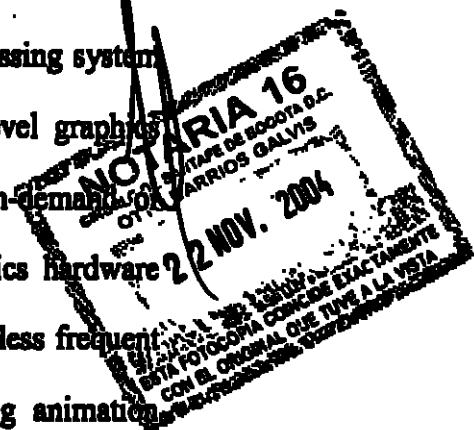


589

1 Via the interfaces, the MIL provides access to a data structure for storing
2 visual information so that applications can take advantage of the graphics
3 capabilities provided by the computer hardware. The interfaces are based on an
4 element object model and a vector graphics markup language for using that
5 element object model in a manner that allows program code developers to
6 consistently interface with a scene graph data structure to produce graphics. The
7 data structure may also be used for either directly rendering or for "compiling" the
8 visual information so that it can be provided to a lower level graphics system for
9 fast composition and animation.

10 The vector graphics element object model generally corresponds to shape
11 elements and other elements including image and video elements that correlate
12 with a scene graph object model of the scene graph. Markup may be parsed into
13 data including elements in an element tree that is translated into the objects of a
14 scene graph data structure. Other markup may be translated directly into data and
15 calls that create the scene graph objects. The markup language provides distinct
16 ways to describe an element, including a simple string format or complex property
17 syntax, which may be named, enabling reuse in other locations in the markup.

18 An aspect of the MIL is the integration of animation and timing across the
19 API set, providing animation as an inherent base-level concept. To facilitate
20 smooth animation, the MIL provides a multiple-level graphics processing system
21 and method (e.g., of an operating system). One such multiple-level graphics
22 processing system comprises two components, including a tick-on-demand or
23 slow-tick high-level component, and a fast-tick (e.g., at the graphics hardware
24 frame refresh rate) low-level component. In general, the high-level, less frequent
25 component performs computationally intensive aspects of updating animation.



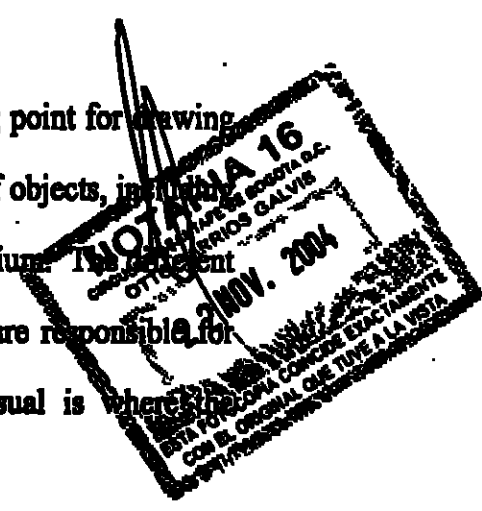
87

580
590

1 parameters and traversing scene data structures, in order to pass simplified data
2 structures to the low-level component. The low-level component operates at a
3 higher frequency, such as the frame refresh rate of the graphics subsystem, to
4 process the data structures into constant output data for the graphics subsystem.
5 The low-level processing includes interpolating any parameter intervals as
6 necessary to obtain instantaneous values to render the scene for each frame of
7 animation.

8 Top level MIL objects include a visual tree, which is an object that contains
9 the main content to be drawn. Controls will derive from visuals of the tree
10 directly. Visuals are device and parent context independent. A render target is the
11 device to which the visual is drawn. This object (e.g., screen) may have its own
12 dirty or invalidation mechanism. Various render targets include a screen in a
13 window, a Printer, a Metafile, a Surface, and a "Sub-window" which is a part of
14 the scene that is drawn separately from the rest of the scene. Other drawing
15 related objects include a Visual Renderer, comprising an object that is configured
16 to draw a visual tree onto a render target, and a Display Scheduler object that
17 knows when to draw the visual tree on to the render target. A Time Manager is a
18 context object for a set of timing nodes, and is the object that the scheduler calls
19 tick on.

20 A Visual API is provided, which is essentially a starting point for drawing
21 via the media integration layer, and comprises multiple types of objects, including
22 a VisualManager object, which connects a Visual Tree to a medium. The different
23 types of VisualManagers (e.g., Screen, Printer, and Surface) are responsible for
24 rendering a Visual Tree to their particular medium. A visual is where the
25



88

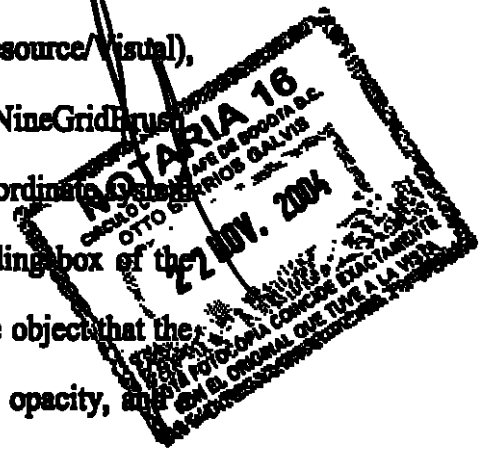
591

1 programmer does the drawing; it is a node in the visual tree and provides a place
2 for a program to draw.

3 The DrawingContext APIs present a context-based programming model for
4 how to construct visual content that populates a Visual or are rendered to an
5 ImageData. DrawingContext classes are provided, as well as the classes and
6 entrypoints necessary to acquire a DrawingContext and enumerate the visual
7 content in a RetainedVisual/DrawingVisual.

8 To enable mutability, there is provided a single set of types that derive from
9 a common Changeable base class. Any type for which mutability is desired
10 change may derive from the Changeable class. For example, in a graphics
11 programming, the object model includes Brushes, Pens, Geometries,
12 FloatAnimations, GradientStops, Segments, and so forth. An IsChangeable
13 property specifies whether the changeable object can be modified or not,
14 depending on its current value, which defines a state.

15 A brush is an object that represents a method to fill a plane. In addition to
16 being able to fill a plane in an absolute way, brushes of the media integration layer
17 are also able to adapt how they fill the plane relative to the size of the object that
18 they are filling. Examples of types of brushes include SolidColorBrush,
19 VisualBrush (which can reference a vector graphics resource/visual),
20 DrawingBrush, LinearGradient, RadialGradient, ImageBrush and NineGridBrush.
21 Certain brush objects will have an idea of how they relate to the coordinate system
22 when they are used, and an idea of how they relate to the bounding box of the
23 geometry with which they are used. This size is based on from the object that the
24 brush is filling. The Brush base class has a Transform, a general opacity, and a
25 blend mode. Brush (and other object resources in Vector Graphics and the MIL



89

592

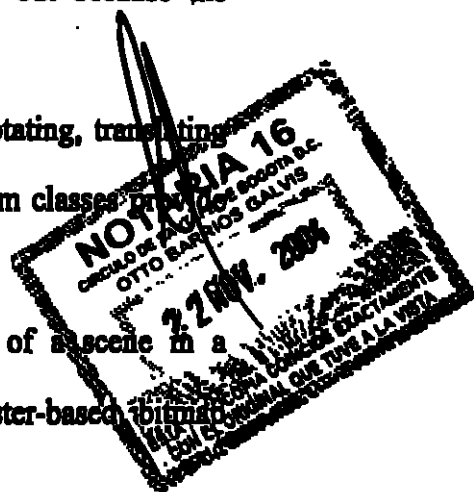
API) objects are Changeables and are writable after they have been created, and follow the general Changeable pattern for how they behave after they are used in qualified use.

A Geometry class of objects can be used for clipping, hit-testing and rendering of 2D vector-based data with the Pen and Brush. The derived Geometry classes provide more specific building and enumeration semantics. A number of shape-specific Geometry types are provided, as well as a generalized PathGeometry that allows for explicit definition of more complex shaped Geometry. Geometry is an abstract base class. A GeometryCollection is a collection of multiple Geometry objects that have been combined using particular CombineMode operations on their defined area. This object allows easier building visual combinations of Geometry objects than is possible using strictly PathFigure objects within a PathGeometry.

ImageSource is an abstract class, comprising a basic building block for imaging. An ImageSource conceptually represents a single, constant set of pixels at a certain size and resolution. For example, an ImageSource may be a single frame in an image file that a Decoder could provide, or it may be the results of a transform that operates on a certain ImageSource of its own. An ImageSource is changeable, not because its own properties can be changed, but because the properties of its sub-classes can potentially be changed.

A Transform class of objects is provided for scaling, rotating, translating and skewing vector and raster graphics. The derived Transform classes provide friendly usage and enumeration semantics.

Effects provide a means to alter the visual contents of a scene in a rendering-centric manner. For example, VisualEffects (raster-based bitmap)

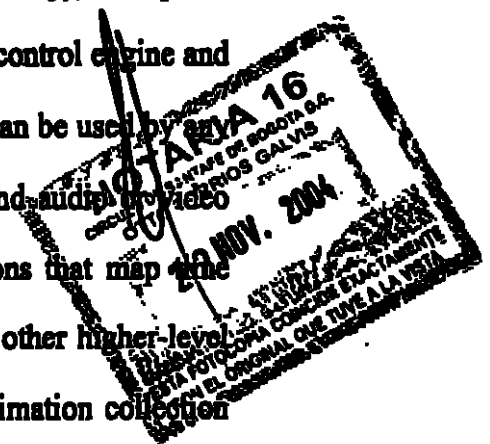


90

effects) operate on the image-based, fully composited representation of a portion of a scene. Effects are broken down into various types including VisualEffects, BlendModes and VectorEffects. VisualEffects can be used in a retained-mode scene by applying it to a sub-graph or an Element, or it can be used in the standalone image pipeline. BlendModes are a specific form of image-based effects, and can be applied to the retained mode scene in generally the same manner as VisualEffects. Blend modes perform a combination of the source and destination colors as the source is composited, e.g., multiply or add.

Hit testing is used to pick visuals in a scene, and operates by starting from the top of the control tree, and returning a control or set of controls by a point or geometry. A control can define whether it is hit or not with support services including rendered geometry, bounding box, out-of-band geometry (hit region), image opacity or mask, and its own logic. The control can return specific hit-related data on hit. The hit test mechanism can filter hit test results in an efficient manner. The hit test walk is a deep right to left walk of the visual tree, with hits reported through a callback in z-order, top-to-bottom fashion. When descending, the hit tester views the filtering in terms of element level relationships, for example, a canvas with shapes, or a dock panel with an inner canvas. When a hit occurs, the hit tester can either continue processing further hits (if any), or stop.

An animation system is provided, comprised of a timing control engine and a set of animation objects. The timing engine is a service that can be used by objects that exhibit time-varying behaviors, e.g., animations and audio/video media objects. Animation objects implement a set of functions that map the spans into other data types, which are then used as inputs into other higher-level objects. Graphical animation is achieved by associating an animation collection



594

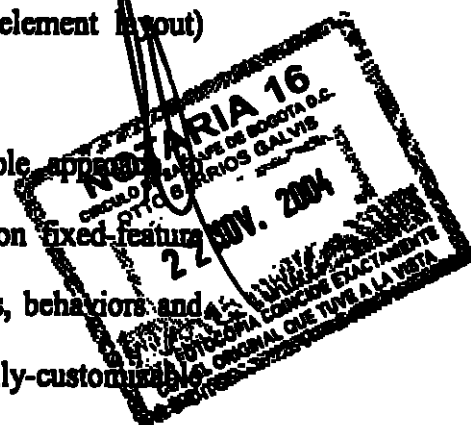
1 with a rendering operation. Each animation used in a rendering operation may be
2 run on a separate clock, referred to as a called a "timeline." Once an animated
3 primitive is drawn and animation parameters specified, the low-level rendering
4 system takes care of redrawing the scene at regular intervals. Each time a frame is
5 rendered the current value of the animations involved in the scene is computed,
6 based on the elapsed time (in most cases measured by the system clock), and then
7 the animated primitives are redrawn.

8 Various primitive types, color features and media support is also provided
9 via the MIL. MediaData can be used to play any audio/video content.

12 Design Namespace

13 A design namespace 320 provides classes that enable the editing of forms
14 and text, formatting data and cross-process data sharing. These classes provide an
15 extensible framework for editing documents, applications, and other content.
16 Design namespace 320 contains functionality on two levels: high-level
17 functionality for application developers desiring pre-packaged ready-to-use editors
18 for different types of information; and lower-level functionality for more advanced
19 applications introducing its own types of data. Example pre-packaged ready-to-
20 use editors offer plain text editing, rich text editing, forms (element layout)
21 editing, and password input.

22 Design namespace 320 provides a flexible and extensible apparatus
23 organizing various functionality. Instead of providing a common fixed-feature
24 editor, design namespace 320 provides a combination of services, behaviors and
25 abstractions that allow developers to build specialized and easily-customizable



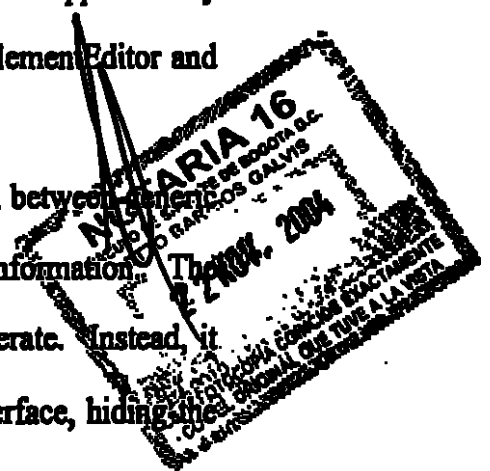
f2

information editing solutions. Design namespace 320 includes several customizable editing features, including: scoped services, stackable behaviors, editor-designer pattern, abstract designer for element editing, abstract text object model, adorning layer, design surface, generalized data transfer protocol, and extensible undo mechanism.

Scoped services allows the association of specific services with particular portions and sub-portions of application data. This mechanism also allows turning services on and off in different scopes. The main class providing a support for services is ServiceManager; individual services are implemented using IService and IScopedService interfaces.

The concept of "stackable behaviors" allows different behaviors to be activated in appropriate periods of time. In one embodiment, time-based behavior activation is nested, such that a sub-process can be started and finished while a more general process is temporary suspended and then properly restored. Rather than hard-coding a solution, the approach of stackable behavior solves this problem by allowing an integration of processes that do not have pre-defined knowledge about each other. The main class providing support for this mechanism is EditRouter; individual behaviors are subclasses of Editor and EditBehavior classes. Built-in types of editable data are supported by corresponding subclasses of EditBehavior such as TextEditor, ElementEditor and MoveBehavior.

The concept of "editor-designer pattern" allows separation between generic editing functionality and more specific sub-typed of editable information. The editor does not expect a specific data structure on which to operate. Instead, it assumes that data content is exposed through some abstract interface, hiding the

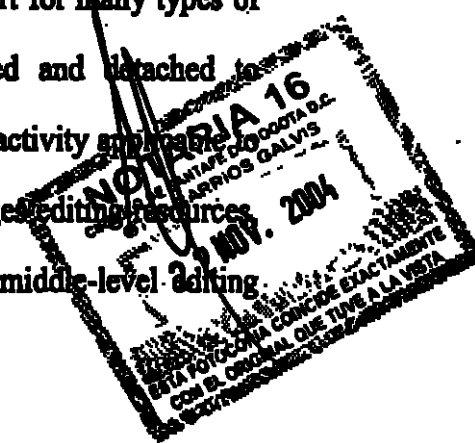


1 implementation details, but sufficient for corresponding action of editing
2 interaction. This pattern is particularly useful with elements (in forms editing) and
3 for text (in rich text editing).

4 "Abstract designer for element editing" is an application of the "editor-
5 designer pattern" for an area of element editing (e.g., forms editing). Various
6 elements may have different internal structures, appearances and behaviors, but if
7 they expose some standard "designers" for movement, rotation, or resizing, then
8 they become editable by the element editor.

9 The abstract text model is another application of the "editor-designer
10 pattern" for use in rich text editing. This abstraction allows different backing
11 stores to participate in rich text editing, such as RTF, XHTML, plain text, XAML
12 markup, syntactically highlighted source code, etc. Even though the semantic
13 nature and internal structure of these types of texts can be different, by exposing
14 itself via an abstract text model they allow the application of generic editing
15 mechanisms. Additionally, this approach allows for text and data integration, such
16 that text from different backing stores may contain pieces of each other, while
17 providing a seamless editing experience.

18 The adorning layer is an unified tool that provides rich visual feedback
19 during editing. The adorning layer provides uniformed support for many types of
20 feedback. Powerful visuals may be dynamically attached and detached to
21 elements and other portions of data to indicate potential user activity and
22 them. The design surface is a XAML control that pre-packages editing resources
23 such as the adorning layer and edit router. This simplifies middle-level editing
24 development.
25



597

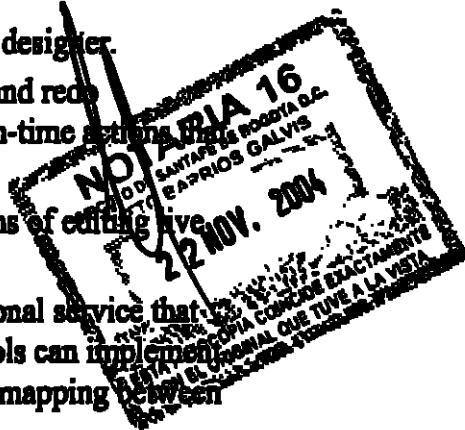
Editing functions such as cut/copy/paste and drag/drop are important in an integrated information environment, but can be difficult to support due to the complexity and incompatibility of different data types. The generalized data transfer protocol addresses this difficulty by providing for the combination of abstractions used for data extraction, data insertion, data dragging, and data conversion. This generalized data transfer protocol provides a powerful and flexible mechanism for application-data integration.

The extensible undo mechanism manages an undo stack by collecting data of various types. The undo manager includes protocols to handle undo operations with different data types.

The following tables list example members exposed by the System.Windows.Design namespace.

Classes

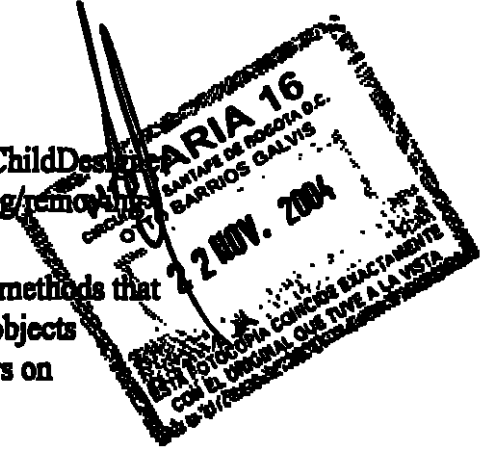
ActivationEventFilter	EventFilter
ActivationRectangleAdornerInfo	This is the feedback that shows the border on DesignActive elements
AdornerInfo	Provides information about a specific adorner in an adornerset.
AdornerLayer	Provides a surface for displaying decorations on elements that need to transcend Z-order (the elements always need to be on top).
CanvasDesigner	Default Canvas designer.
ComponentChangeService	Provides undo and redo functionality for design-time actions that affect components.
Designer	Provides a means of editing live content.
DesignerFilterService	Defines an optional service that applications and controls can implement to override the default mapping between



95

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

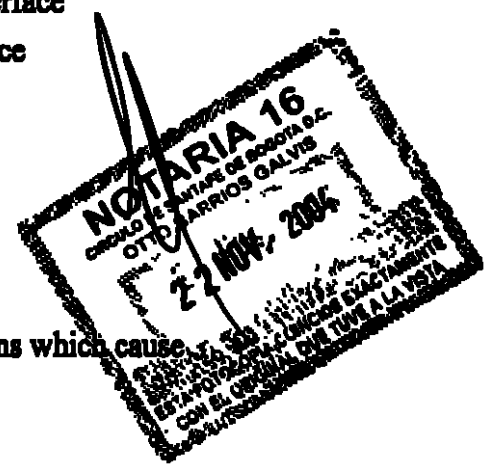
	Editor).
SelectionService	Provides programmatic access to selection of UI items. You can get feedback on selected items, as well as change selected items and their properties.
ServiceConverter	ServiceConverter - Converter class for converting between a string and an instance of IService.
SimpleTextDesigner	ITextDesigner implementation for Text and other controls that support ITextParagraphResult.
TextDecorator	TextDecorator defines an object that manages text decoration from competing services. Using TextDecorator instead of writing property values privately ensures that competing decoration will never overlap, even in the case of identical priority.
TextEditor	Provides text selection services.
TextSelection	The TextSelection class encapsulates selection state for the TextEditor class. It has no public constructor, but is exposed via a public property on TextEditor.
UndoService	Framework implementation of IUndoService
UndoStackChangedEventArgs	Provides data for the UndoStackChanged and RedoStackChanged events
Interfaces	
IAddRemoveChildDesigner	The IAddRemoveChildDesigner interface is used for adding/removing children
IAdornerDesigner	Provides access to methods that allow you to retrieve the objects necessary to draw adorners on FrameworkElements.



590
600

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

	The inner element in DesignActivation that can become designActive. Has the name container as it is assumed that if an element is DesignActive, then it will most likely be editing some child content
IDesignActivationContainer	
	An IDesignActivationContainerParent can be host IDesignActivationContainer's in a Design environment
IDesignActivationContainerParent	
IDesignSurface	Interface to be implemented by any element wishing to participate in editing.
IMoveDesigner	Move designer interface
IParentUndoUnit	IParentUndoUnit interface
IScopedService	IScopedService
ISelection	base interface for selection modes
	Designer interface associated with the SelectionBehavior (SelectionRouter). A class that implements this would be specified in the Designer Attribute of an element type
ISelectionDesigner	
IService	This is a placeholder for a TypeConverter for editing services.
	An interface components implement to supply the TextEditor service with layout-specific information and actions.
ITextDesigner	
IUndoService	IUndoService interface
IUndoUnit	IUndoUnit interface
Enumerations	
AdornerInfoZOrder	Z-order flags
AdornerRenderTriggers	Flags indicating the conditions which cause adorners to rerender

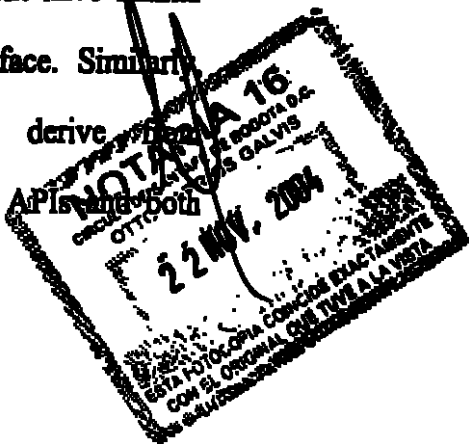


78

1	EditRouterChange	Enumeration of possible router change actions
2	TextDecorationPriority	TextDecorationPriority ranks the relative importance of various text markup. It is used to z-order decoration runs in the TextDecorator class.
3	UndoState	Enum for the state of the undo manager
4		
5		
6	Delegates	
7		
8	UndoStackChangedEventHandler	Represents the method that will handle an UndoStackChanged or RedoStackChanged event.
9		
10		

11 Input Namespace

12 An input namespace 322 includes an input manager that coordinates inputs
 13 received by the system. The input namespace 322 also includes classes that help
 14 manage and provide control for different input devices, such as a keyboard or a
 15 mouse. The input system allows applications to get information from input
 16 devices such as mice, keyboards, and styluses (pens). Most input functionality is
 17 in the form of properties, methods, and events on the UIElement,
 18 FrameworkElement, ContentElement, and ContentFrameworkElement classes in
 19 the System.Windows namespace. UIElement and ContentElement have similar
 20 input functionality, which is captured by the IInputElement interface. Similarly,
 21 FrameworkElement and ContentFrameworkElement, which derive
 22 UIElement and ContentElement respectively, share many input APIs and both
 23 implement IFrameworkInputElement.
 24
 25



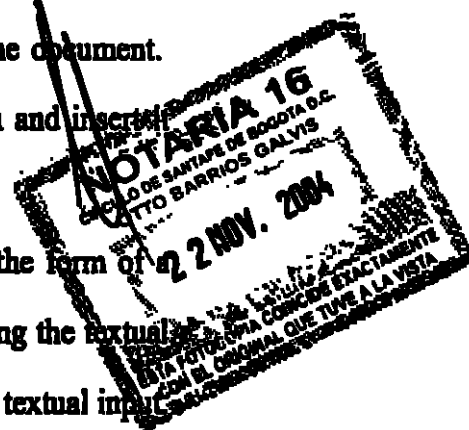
1 The input system provides full support for keyboards, mouse, and styles.

2 Keyboard functionality includes keydown/up events, focus management and
3 notification, accessing current key state, and converting keystrokes into input
4 strings. Mouse functionality includes mouse position, clicks, movement,
5 enter/leave events at element boundaries, mouse capture, hover and mouse wheel.
6 Stylus functionality includes position and movement (both when the pen is
7 touching the surface and when it is "in air"), taps/clicks, capture, and gesture
8 recognition.

9 The Input namespace contains helper classes necessary for the above
10 functionality, such as enumerations, events arguments, delegate signatures, etc.
11 Additionally, the Input namespace includes the Keyboard, Mouse, and Stylus
12 classes, which provide information relating to the current state of those devices.
13 The input system provides ways of specifying mouse cursors on a particular
14 element, and for an entire application.

15 The input system is integrated with Text Services Framework (TSF), which
16 allows other input software to understand the textual context into which new text
17 is being input. This is used by input method editors (IMEs), which allow users of
18 East Asian languages to turn multiple keystrokes into a single character -- the IME
19 software communicates with TSF to identify the best interpretation of the
20 keystrokes based on context and uses TSF to insert the text into the document.
21 Similarly, speech recognition uses TSF to pick the best recognition and insert
22 into the document.

23 The input system offers automatic integration with TSF, in the form of
24 TextInput event. The Input namespace also offers APIs for describing the textual
25 context, and controlling the conversion from raw keyboard input into textual input.



The InputManager class provides filters and monitors, which allow a third party to observe the input stream and create new input events before the original input events have been fired. The InputManager also provides APIs for new devices to be plugged into the input system, using InputProvider and InputReport.

The following tables list example members exposed by the System.Windows.Input namespace.

Classes

AccessKeyManager

AccessKeyManager exposes access key functionality, also known as mnemonics. A typical example is the Ok button, with alt-O being the mnemonic – pressing alt-O is the same as clicking the button.

Cursor

Represents the mouse cursor to use.

CursorTypeConverter

Converts mouse cursors to and from strings.

FocusChangedEventArgs

The FocusChangedEventArgs class contains information about focus changes.

InputDevice

Provides the base class for all input devices.

InputEventArgs

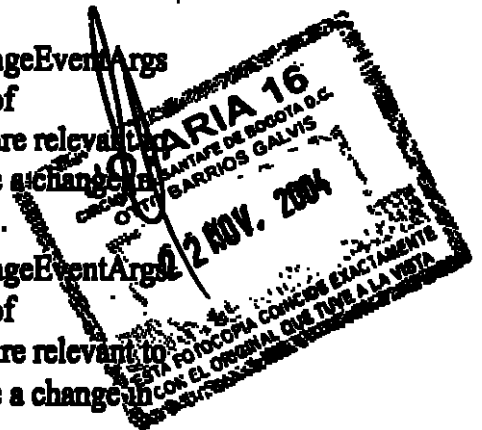
The InputEventArgs class represents a type of RoutedEventArgs that are relevant to all input events.

InputLanguageChangedEventArgs

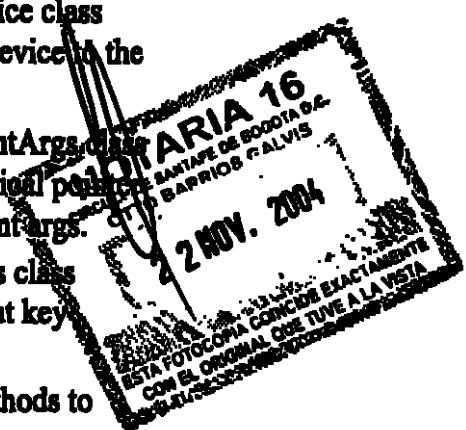
The InputLanguageEventArgs class represents a type of RoutedEventArgs that are relevant to events raised to indicate a change in (human) input language.

InputLanguageChangingEventArgs

The InputLanguageEventArgs class represents a type of RoutedEventArgs that are relevant to events raised to indicate a change in



	(human) input language.
1	
2	The InputLanguageEventArgs
3	class represents a type of
4	RoutedEventArgs that are relevant to
5	events raised to indicate a change in
6	(human) input language.
7	
8	The InputLanguageManager
9	class is responsible for managing
10	the input language in Avalon.
11	
12	The InputManager class is
13	responsible for coordinating all of the
14	input system in "Avalon".
15	
16	The InputMethod class
17	exposes TSF-related APIs, which are
18	communicating or accessing TIP's
19	properties.
20	
21	This
22	InputMethodStateChangedEventArgs
23	class is used when the input method
24	editor (IME) changes its state.
25	
	The object which input
	providers use to report input to the
	input manager.
	The InputReport is an abstract
	base class for all input that is
	reported to the InputManager.
	Provides data for the event
	that is raised when an InputReport is
	being processed.
	The Keyboard class exposes
	APIs related to the keyboard.
	The KeyboardDevice class
	represents the keyboard device to the
	members of a context.
	The KeyboardEventArgs class
	provides access to the logical position
	device for all derived event args.
	The KeyEventArgs class
	contains information about key
	states.
	Provides static methods to



605

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Mouse

convert between Win32 VirtualKeys and the "Avalon" Key enum.

The Mouse class exposes APIs related to the mouse.

MouseButtonEventArgs

The MouseButtonEventArgs describes the state of a Mouse button.

MouseDevice

The MouseDevice class represents the mouse device to the members of a context.

MouseDoubleClickEventArgs

Provides data for events that are raised when the mouse is doubled-clicked.

MouseEventArgs

The MouseEventArgs class provides access to the logical Mouse device for all derived event args.

MouseWheelEventArgs

The MouseWheelEventArgs describes the state of a Mouse wheel.

NotifyInputEventArgs

Provides information about an input event being processed by the input manager.

PreProcessInputEventArgs

Allows the handler to cancel the processing of an input event.

ProcessInputEventArgs

Provides access to the input manager's staging area.

RawKeyboardInputReport

The RawKeyboardInputReport class encapsulates the raw input provided from a keyboard.

RawMouseInputReport

The RawMouseInputReport class encapsulates the raw input provided from a mouse.

StagingAreaInputItem

This class encapsulates an input event while it is being processed by the input manager.

TextInputEventArgs

The TextInputEventArgs contains a text representation of input.

TextManager

The TextManager class provides the input-to-text event promotion.



103

TextServicesContext This class manages interop between a UIDispatcher and the Text Services Framework, a native COM API that enables east-asian IME input.

TextStoreInfo This is an internal, link demand protected class.

Interfaces

IInputLanguageSource An interface for controlling the input language source.

IInputProvider An interface implemented by all input providers.

IKeyboardInputProvider An interface for controlling the keyboard input provider.

IMouseInputProvider An interface for controlling the mouse input provider.

Enumerations

CursorType An enumeration of the supported cursor types.

ImeConversionMode ImeConversionMode

ImeSentenceMode ImeSentenceMode

InputMethodState State of Ime

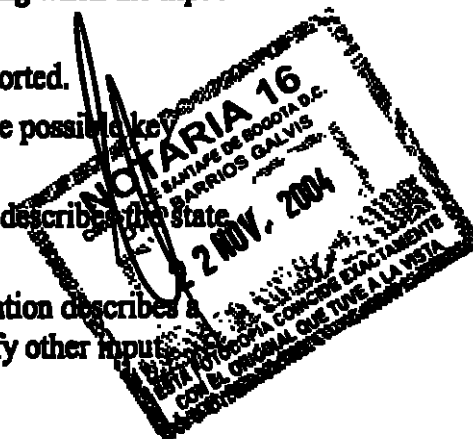
InputMode The mode of input processing when the input was provided.

InputType The type of input being reported.

Key An enumeration of all of the possible key values on a keyboard.

KeyState The KeyState enumeration describes the state that keyboard keys can be in.

ModifierKeys The ModifierKeys enumeration describes a set of common keys used to modify other input operations.



607

MouseButton The MouseButton enumeration describes the buttons available on the mouse device.

MouseButtonState The MouseButtonState enumeration describes the possible states of the buttons available on the Mouse input device.

RawKeyboardActions The raw actions being reported from the keyboard.

RawMouseActions The raw actions being reported from the mouse.

SpeechMode Mode of speech

Structures

TextServicesMSG Managed version of the Win32 MSG struct.

Delegates

FocusChangedEventHandler The delegate to use for handlers that receive FocusChangedEventArgs.

InputEventHandler The delegate to use for handlers that receive InputEventArgs.

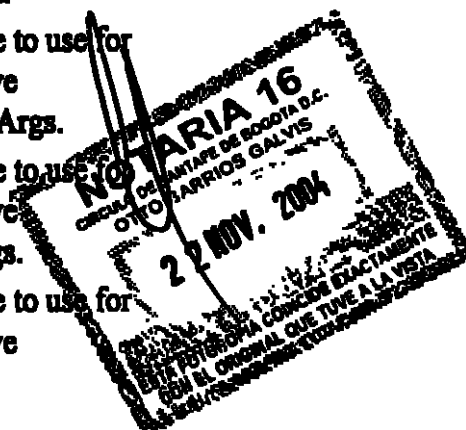
InputLanguageEventHandler This is a delegate for InputLanguageChanged and InputLanguageChanging events.

InputMethodStateChangedEventHandler The delegate to use for handlers that receive input method state changed event.

InputReportEventHandler The delegate to use for handlers that receive PointerMoveEventArgs.

KeyboardEventHandler The delegate to use for handlers that receive KeyboardEventArgs.

KeyEventHandler The delegate to use for handlers that receive



105

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

MouseButtonEventHandler

KeyEventArgs.

The delegate to use for handlers that receive MouseButtonEventArgs.

MouseDoubleClickEventHandler

The delegate to use for handlers that receive MouseDoubleClickEventArgs.

MouseEventHandler

The delegate to use for handlers that receive MouseEventArgs.

MouseWheelEventHandler

The delegate to use for handlers that receive MouseWheelEventArgs.

NotifyInputEventHandler

Delegate type for handles of events that use NotifyInputEventArgs.

PreProcessInputEventHandler

Delegate type for handles of events that use PreProcessInputEventArgs.

ProcessInputEventHandler

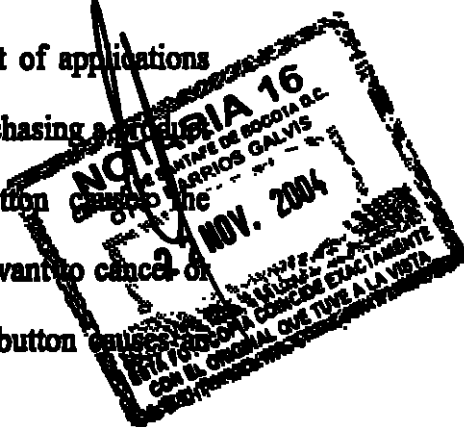
Delegate type for handles of events that use ProcessInputEventArgs.

TextInputEventHandler

The delegate to use for handlers that receive TextInputEventArgs.

Navigation Namespace

A navigation namespace 324 provides a set of classes and services that allow the building of applications with navigation paradigms, such as a browser application. These classes and services permit the development of applications with customized navigation experiences. For example, when purchasing a product or service from an online merchant, clicking a "Back" button causes the application to display a different page that asks the user if they want to cancel or change their order. In another example, activating a "Refresh" button causes



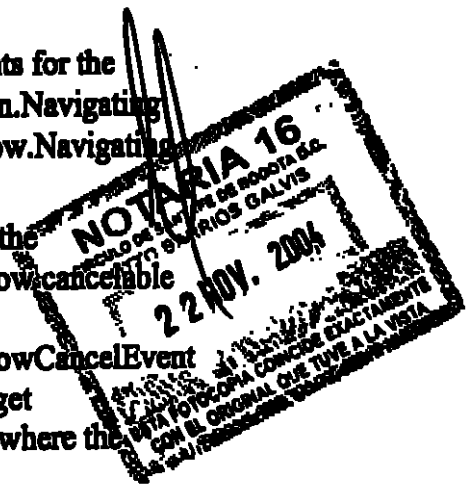
application to retrieve new data instead of first reloading the application followed by retrieving the new data. The navigation namespace 324 also includes page functions that provide a mechanism for generating a hierarchy of questions that are presented to a user.

The following tables list example members exposed by the System.Windows.Navigation namespace.

Classes

BoolWrapper	A wrapper for a Boolean value.
BoolWrapperPageFunction	A typed page function that returns a Boolean value to the previous page.
BoolWrapperReturnEventArgs	
IntWrapper	A wrapper for a Int32 value.
IntWrapperPageFunction	A typed page function that returns a Int32 value to the previous page.
IntWrapperReturnEventArgs	
Journal	Contains an application's navigation history.
JournalEntry	Represents a journal entry.
LoaderService	Used to set the current loader in a given appdomain.
NavigateEventArgs	Obsolete.
NavigatingCancelEventArgs	Event arguments for the NavigationApplication.Navigating and NavigationWindow.Navigating events.
NavigatingNewWindowCancelEventArgs	Event args for the NavigatingNewWindowCancel event
NavigatingNewWindowCancelEventArgs	The event The NavigatingNewWindowCancelEventArgs specifies the target NavigationContainer where the

Args



600
610

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

NavigationApplication

navigation will take place with the Uri or element passed in. By default the Cancel property is set to false. Setting Cancel to true will prevent the new window from being opened, and the navigation will not take place.

Represents a "Longhorn" navigation application to the system.

NavigationContainer

A navigable region that can contain an "Avalon" markup tree. Normally, this class is not used directly but can be used as the parent class for a custom implementation.

NavigationErrorCancelEventArgs

Contains the arguments for the NavigationApplication.NavigationError or and NavigationWindow.NavigationError events.

NavigationEventArgs

Event arguments for non-cancellable navigation events, including LoadCompleted, LoadStarted, Navigated, and NavigationStopped.

NavigationProgressEventArgs

Contains the arguments for the NavigationApplication.NavigationProcess and NavigationWindow.NavigationProcesses events.

NavigationService

Contains the delegates used by navigation events, and a dynamic property that contains an INavigator interface.

NavigationWindow

Represents a navigation window.

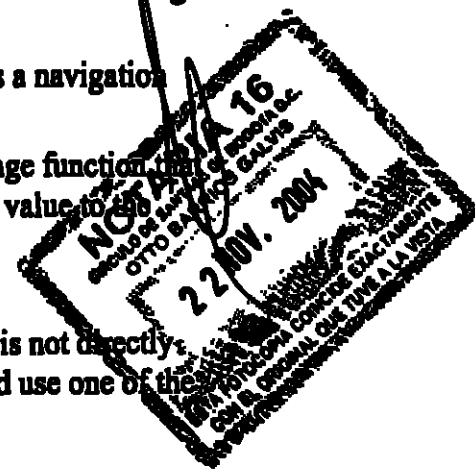
ObjectPageFunction

A typed page function that returns an Object value to the previous page.

ObjectReturnEventArgs

PageFunction

This class is not directly supported. Instead use one of the typed classes:



108

BoolWrapperPageFunction,
IntWrapperPageFunction,
ObjectPageFunction, or
StringPageFunction.

The event argument object for the Return event. This class is not directly supported. Instead use the return arguments for the appropriate typed class:

ReturnArgs

BoolWrapperReturnEventArgs,
IntWrapperReturnEventArgs,
ObjectReturnEventArgs, or
StringReturnEventArgs.

StringPageFunction

A typed page function that returns a String value to the previous page.

StringReturnEventArgs

WindowNavigationContainer

Represents a navigable region within a navigation window.

Interfaces

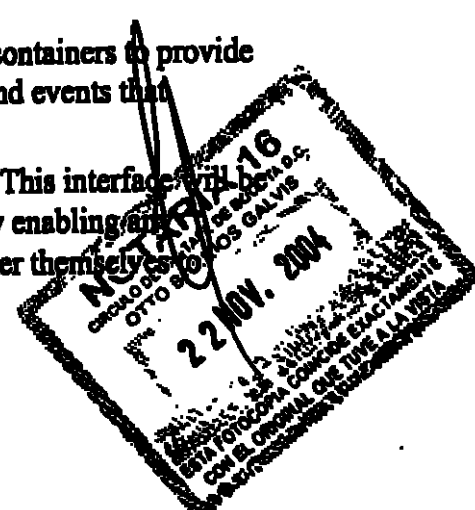
IJournalData **IJournalData** interface - Should to be implemented by Controls that need to persist state in the journal, and restore it when the page is revisited

ILoader Interface used to resolve Uri's to streams. It can be used to load content from file, http, container and managed and unmanaged resources

INavigator Implemented by navigation containers to provide access to the properties, methods, and events that support navigation.

INavigatorService INavigatorService interface. This interface will be available on the NavigationWindow enabling any implementor of INavigator to register themselves to participate in Hyperlink targetting.

Enumerations



612

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

NavigationMode Used to specify the navigation mode.

Delegates

BoolWrapperReturnEventHandler
IntWrapperReturnEventHandler

Represents the method that handles the **NavigationApplication.LoadCompleted** and **NavigationWindow.LoadCompleted** events.

LoadCompletedEventHandler

Represents the method that handles the **LoadStarted** event.

LoadStartedEventHandler

Represents the method that handles the **NavigationApplication.Navigated** and **NavigationWindow.Navigated** events.

NavigatedEventHandler

NavigateEventHandler

Obsolete.

Represents the method that handles the **NavigationApplication.Navigating** and **NavigationWindow.Navigating** events.

NavigatingCancelEventHandler

Represents the method that handles the **NavigatingNewWindow** event.

NavigatingNewWindowCancelEventHandler

This delegate is used with the **NavigationApplication.NavigationError** and **NavigationWindow.NavigationError** events.

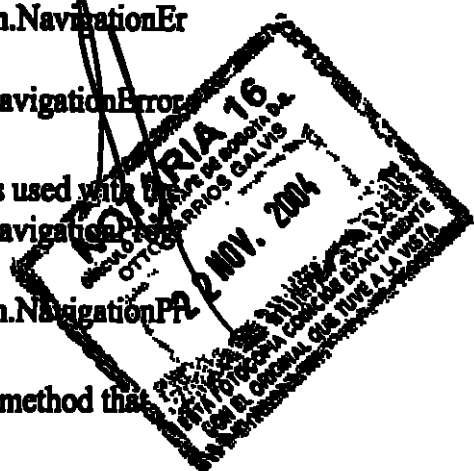
NavigationErrorCancelEventHandler

This delegate is used with the **NavigationWindow.NavigationProgress** and **NavigationApplication.NavigationProgress** events.

NavigationProgressEventHandler

NavigationStoppedEventHandler

Represents the method that



112

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

handles the
NavigationApplication.NavigationSt
opped and
NavigationWindow.NavigationStopp
ed events.

ObjectReturnEventHandler

Represents the method that
handles the Return event. This class
is not directly supported. Instead use
the return event handler for the
appropriate typed class:
BoolWrapperReturnEventHandler,
IntWrapperReturnEventHandler,
ObjectReturnEventHandler, or
StringReturnEventHandler.

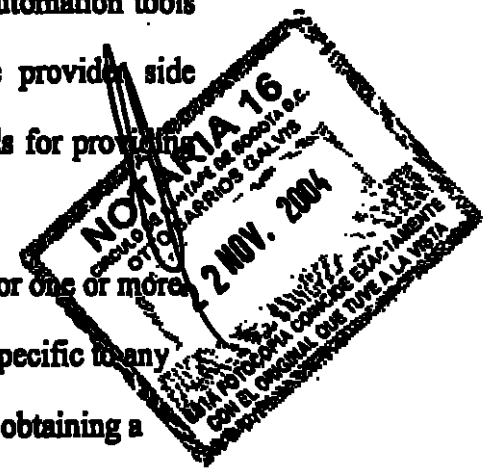
ReturnEventHandler

StringReturnEventHandler

Automation Namespace

An automation namespace 326 contains members used to support
accessibility and user interface automation. The accessibility system includes a
client side and a provider side. A system of tools includes client side automation
tools including a client automation class for seeking user interface information.
The client automation class includes event registration tools and logical element
discovery tools. The set of tools further includes provider side automation tools
for providing the client with user interface information. The provider side
automation tools include an automation provider class having tools for providing
the client with event information.

A client automation class provides UI automation methods for one or more
clients. The client automation class contains methods that are not specific to any
UI element. The client automation class may provide a method for obtaining a



64

1 logical or raw element from a point, a window handle, or the desktop root element.
2 The client automation class additionally may provide methods for finding a logical
3 element based input criteria. The client automation class preferably also includes
4 a method for registering and un-registering for event notifications. The
5 automation class preferably also provides helper functions for loading proxy
6 DLLs, retrieving the localized names of properties and control patterns and
7 performing element comparisons. The client automation class also includes
8 methods for clients to listen for events. The following tables list example
9 members exposed by the System.Windows.Automation namespace.
10

11 **Classes**

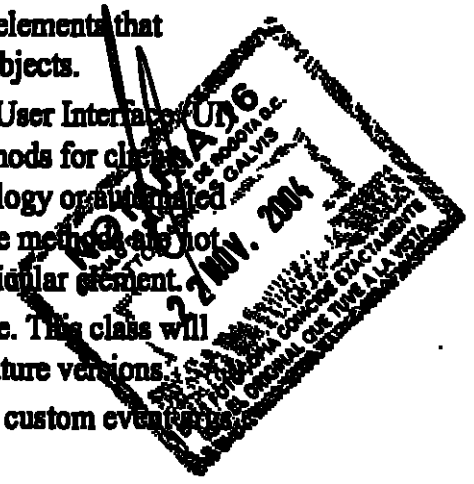
12		Exposes the behavior and
13		information typically associated
14		with a top-level application
15	ApplicationWindowPattern	window. Clients can use this class
16		to tile or cascade the application's
17		multiple document interface (MDI)
18		children, find its button on the
19		taskbar, and locate well-known
20		sections of its user interface such as
21		toolbars and menus.

22	AssociatedInformationPattern	Exposes semantic and
23		metadata for UI elements that
24		represent other objects.

25	Automation	Contains User Interface (UI) 6
26		Automation methods for clients (assistive technology or automated
27		test script). These methods are not
28		specific to a particular element.

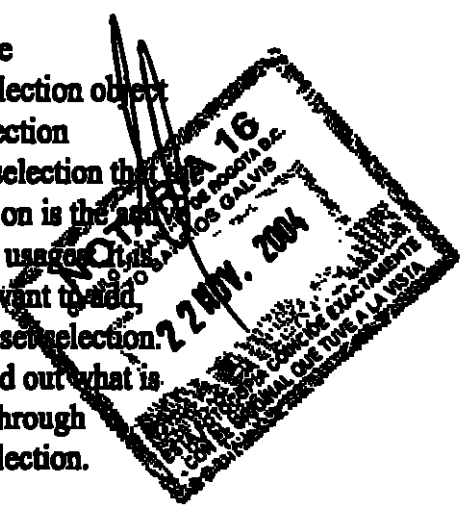
29	AutomationEvent	Do not use. This class will
30		be removed in future versions.

31	AutomationEventArgs	Pattern or custom event args
32		class

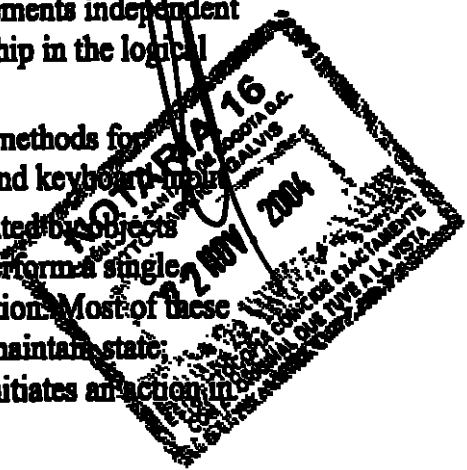


112

1	AutomationFocusChangedEventArgs	Focus event args class
2	AutomationIdentifier	Base class for object identity-based identifiers. This class is effectively abstract; only derived classes are instantiated.
3		
4	AutomationIdentifierProxy	Internal class used to deserialize AutomationIdentifiers. Should not be used directly.
5		
6	AutomationPattern	Do not use. This class will be removed in future versions.
7	AutomationPermission	Provides a set of permissions for accessing UI elements. This class cannot be inherited.
8		
9	AutomationPermissionAttribute	Provides methods and properties for AutomationPermissionAttribute. This class cannot be inherited.
10		
11	AutomationProperty	Do not use. This class will be removed in future versions.
12	AutomationPropertyChangedEventArgs	PropertyChanged event args class
13	AutomationTextAttribute	Identifier for Automation Text Attributes
14		
15	AutomationTextPointer	Represents the position of a character within text. AutomationTextPointer provides methods and properties for accessing text and text navigation.
16		
17	AutomationTextRange	Used to get, set, add and remove selection.
18		
19		
20		
21		
22	AutomationTextSelection	Purpose: The AutomationTextSelection object handles all text selection management. The selection that the insertion pointer is on is the selection. Example usage: This is used when clients want to add, remove, modify or select selection. Clients can also find out what is currently selected through AutomationTextSelection.
23		
24		
25		

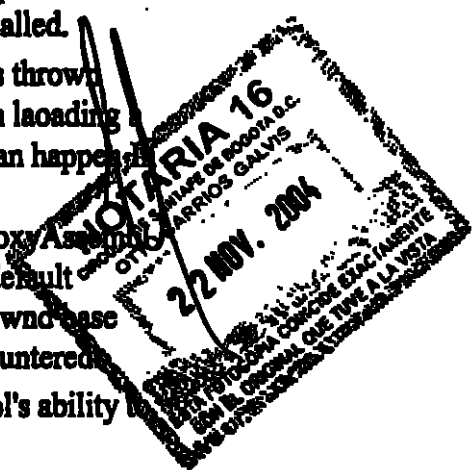


1	AutomationTextUnit	Identifier for Automation Text Units
2	BasePattern	Internal class
3	DockPattern	Expose an element's ability to change its dock state at run time.
4		This exception is thrown
5	ElementNotEnabledException	when client code attempts to manipulate an element or control that is currently not enabled.
6		ElementPath provides
7		criteria necessary to subsequently
8	ElementPath	return to the logical element previously recorded, modified, or wholly created by the application provider.
9		Exposes a control's ability to
10		expand to display more content or to collapse to hide content.
11	ExpandCollapsePattern	Examples include menu buttons, Start button, tree-view item in Windows Explorer, and combo boxes.
12		Exposes the elements of a
13		grid. Allows clients to quickly determine the coordinates of a grid item.
14	GridItemPattern	
15		Exposes basic grid
16	GridPattern	functionality, size, and navigation to specified cells.
17		Exposes the hierarchical
18		relationship between a control's user interface elements independent of their relationship in the logical tree.
19	HierarchyItemPattern	
20		Provides methods for
21	Input	sending mouse and keyboard input.
22		Implemented by objects
23		(controls) that perform a single, unambiguous action. Most of these controls do not maintain state; invoking them initiates an action in
24	InvokePattern	
25		

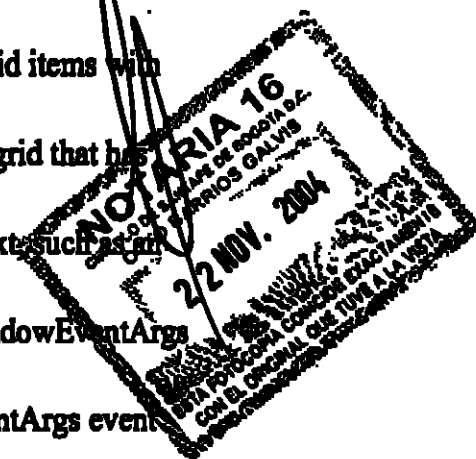


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

	the application. Examples of controls that implement this interface include push buttons, toolbar buttons, menu items, hyperlinks, check boxes, radio buttons, the plus symbol in the tree-view control, and list-view items in Windows Explorer.
LogicalElement	Represents a UI element in the logical tree regardless of implementation ("Avalon", Microsoft® Win32®).
LogicalStructureChangedEventArgs	Logical structure changed event args class.
MultipleViewPattern	Exposes an element's ability to switch between multiple representations of the same set of information, data, or children. Examples include ListView (Thumbnails, Tiles, Icons, List, Details) Excel charts (Pie, Line, Bar, Cell Value with a formula), and Outlook Calendar (Year, Month, Week, Day).
NoClickablePointException	The exception that is thrown when the value of an argument is outside of the allowable range of values, as defined by GetClickablePoint. For example, when the bounding rectangle is empty, has no width or height or the LogicalElement at that point is not the same one that was called.
ProxyAssemblyNotLoadedException	This exception is thrown when there is a problem loading a proxy assembly. This can happen in response to Automation.RegisterProxyAssembly or when loading the default proxies when the first hwnd base LogicalElement is encountered.
RangeValuePattern	Exposes a control's ability



1		manage a value within a finite range. It conveys a control's valid minimum and maximum values and current value.
2		
3	RawElement	Represents an element in the raw element tree.
4		
5	ScrollPattern	Represents UI elements that are expressing a value
6		
7		Represents individual items in containers that manage selection. UI Automation clients use this class to get information about, or to manipulate, UI elements that support the SelectionItemPattern control pattern. Examples include any items that can be selected in a control that supports the SelectionPattern control pattern, such as an item in a list box or tree view.
8		
9	SelectionItemPattern	
10		
11		
12		
13		Represents containers that manage selection. Used by UI Automation clients to get information about, or to manipulate, user interface elements that support the SelectionPattern control pattern.
14	SelectionPattern	
15		
16		Exposes a container's current sort order and allows clients to programmatically re-sort elements.
17	SortPattern	
18		
19	SplitPattern	Represents windows that can clone themselves by creating an adjacent window.
20		
21	TableItemPattern	Represents grid items with header information.
22		
23	TablePattern	Represents a grid that has header information.
24		
25	TextPattern	Represents text, such as an edit control.
	TopLevelWindowEventArgs	TopLevelWindowEventArgs event args class
	TreeLoadEventArgs	TreeLoadEventArgs event



619

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ValuePattern

args class

Exposes values for controls that do not span a range such as radio buttons, toggle buttons, check boxes, edit boxes, RGB color values, and checkable menu items.

VisualInformationPattern

Exposes information about an image or an animation that conveys information to the user such as Internet Explorer's waving flag or spinning globe that indicates a document is downloading or Windows Explorer's flying papers that indicate copying is in progress.

WindowPattern

Exposes an element's ability to change its on-screen position or size. Examples include top-level application windows (Word or Windows Explorer), Outlook's main window panes (folders, e-mail messages, tasks), and MDI child windows.

ZoomPattern

Exposes the current zoom level in a control and allows clients to change it programmatically.

Enumerations

AutomationPermissionFlag

Contains the access flags for an AutomationPermission object.

ChildArrangement

Different ways children may be arranged.

DockPosition

The edge of the container that the dockable window will cling to.

ExpandCollapseState

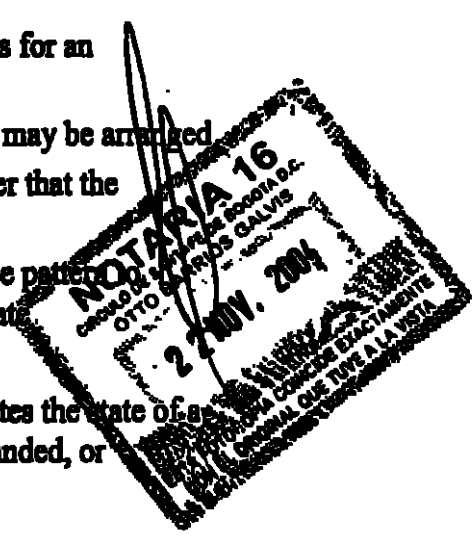
Used by ExpandCollapse pattern to indicate expanded/collapsed state.

GetDataOptions

Options for GetData

HierarchyItemState

Enumeration that indicates the state of a hierarchy item: collapsed, expanded, or partially expanded



157

610

620

1	ItemCheckState	Enumeration that indicates state of checkbox, radio buttons, and similar controls
2	LoadState	Tree state flags.
3	LogicalMapping	Values that indicate whether a node in the raw tree will be displayed in the logical tree
4	ProxyFlags	Enum used to indicate results of requesting a property
5	RowOrColumnMajor	Is the data data in this table best present by row or column
6	ScopeFlags	Flags used to define scope when listening to events
7	ScrollAmount	Used by ScrollPattern to indicate direction (forward or backward) and amount (page or line) by which to scroll
8	SendMouseInputFlags	Flags for SendMouseInput. These flags indicate whether movementnt took place, or whether buttons were pressed or released.
9	SplitStyles	Flag for the directions into which a window may split
10	StructureChangeType	Flags that indicate changes in the structure of the logical tree.
11	TextPointerGravity	Gravity determines the placement of a AutomationTextPointer when text is inserted at the AutomationTextPointer. Gravity is always from the text pointers point of view, i.e. I am before my character or I am after my character. When the user inserts a character at your text pointer location, Before means that your pointer will be placed before the new character, After means that the pointer will be placed After the new character.
12	WindowChangeType	Flags that indicate changes in top-level windows.
13	WindowInteractionState	The current state of the window's interaction
14	WindowVisualState	States used to describe the visual state of the window. WindowVisualState follows the Office and HTML definition of WindowState.

116

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Structures

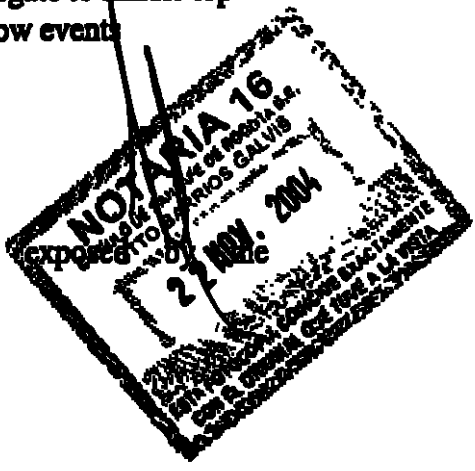
MatchCondition	Specifies criteria for finding elements using FindLogicalElement or FindRawElement.
NameAndData	Contains name, data as an object, and data as a string
ProxyDescription	Structure containing information about a proxy
SortInformation	Information used to sort

Delegates

AutomationEventHandler	Delegate to handle AutomationEvents
AutomationFocusChangedEventHandler	Delegate to handle focus change events
AutomationPropertyChangedEventHandler	Delegate to handle Automation Property change events
LogicalStructureChangedEventHandler	Delegate to handle logical structure change events
ProxyFactoryCallback	Implemented by Hwnd handlers; called by UI Automation framework to request a proxy for specified window. Should return a proxy if supported, or null if not supported.
TopLevelWindowEventHandler	Delegate to handle top level window events

The following tables list example members exposed by the System.Windows.Automation.InteropProvider namespace.

Classes



622

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

AutomationInteropProvider Contains methods used by Win32 applications or controls that implement User Interface (UI) Automation. This class cannot be inherited.

TextPointerPair Represent a contiguous block of characters

Interfaces

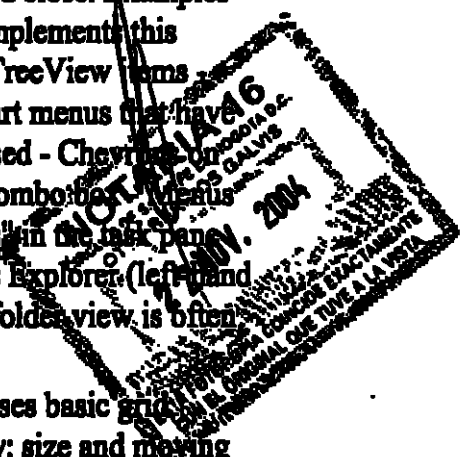
IApplicationWindowInteropProvider Exposes the behavior and information typically associated with a top-level application window.

IAssociatedInformationInteropProvider Exposes semantic and metadata for UI elements that represent other objects.

IDockInteropProvider Expose an element's ability to change its dock state at run time.

IExpandCollapseInteropProvider Exposes a control's ability to expand to display more content or collapse to hide content. Supported in conjunction with the HierarchyItem pattern on TreeView items to provide tree-like behavior, but is also relevant for individual controls that open and close. Examples of UI that implements this includes: - TreeView items - Office's smart menus that have been collapsed - Chevron toolbars - Combo box menus - "Expandos" in the task pane of Windows Explorer (left-hand side where folder view is often displayed).

IGridInteropProvider Exposes basic grid functionality: size and moving



1.20

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

IGridItemInteropProvider

to specified cells.

Represents an item that is within a grid. Has no methods, only properties.

IHierarchyItemInteropProvider

Expose and allow clients to traverse the hierarchical relationship between UI elements independent from their relationship in the logical tree. Hierarchy relationships are by definition noncircular.

Examples of UI that implements this includes: - TreeView items - Visio Org Chart - Menus - Listview controls when "Show in groups" mode is active.

IInvokeInteropProvider

Implemented by objects (controls) that perform a single, unambiguous action. Most of these controls do not maintain state; invoking them initiates an action in the application.

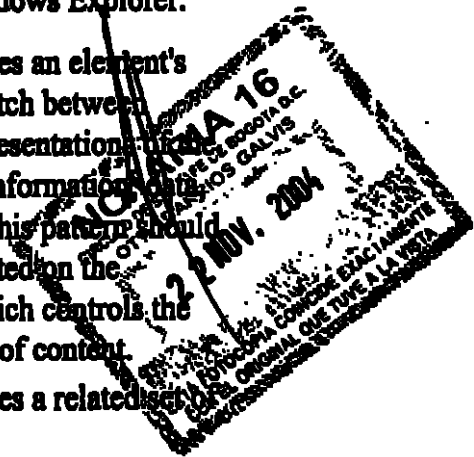
Examples of user interface elements that implement this interface include push buttons, toolbar buttons, menu items, hyperlinks, check boxes, radio buttons, the plus symbol in the tree-view control, and list-view items in Windows Explorer.

IMultipleViewInteropProvider

Exposes an element's ability to switch between multiple representations of the same set of information, or children. This pattern should be implemented on the container which controls the current view of content.

IRangeValueInteropProvider

Exposes a related set



624

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

properties that reflect a controls ability to manage a value within a finite range. It conveys a controls valid minimum and maximum values and its current value. Examples: Numeric Spinners Progress Bar, IP Control (on the individual octets) some Color Pickers ScrollBars some Sliders public interface that represents UI elements that are expressing a current value and a value range. public interface has same definition as IValueProvider. The two patterns' difference is that RangeValue has additional properties, and properties generally do not appear in the pattern public interfaces.

Implemented by providers to expose elements that are part of a structure more than one level deep. For simple one-level structures which have no children, IRawElementProviderSimple can be used instead. The root node of the fragment should support the IRawElementProviderFragmentRoot interface, which is derived from this, and has some additional methods.

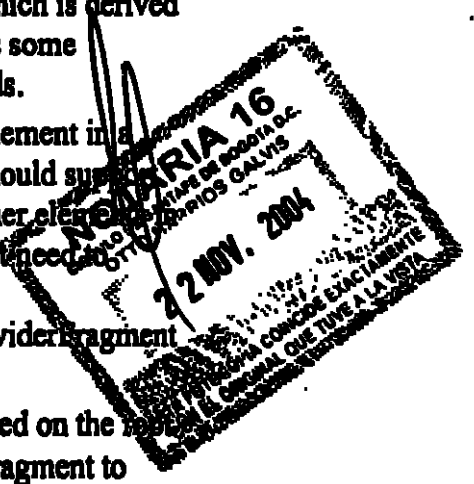
The root element in a fragment of UI should support this interface. Other elements in the same fragment need not support the IRawElementProviderFragment interface.

Implemented on the root element of a UI fragment to

IRawElementProviderFragment

IRawElementProviderFragmentRoot

IRawElementProviderFragmentRootAdviseEvents



122

65

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

IRawElementProviderHwndOverride

allow it to be notified of when it is required to raise automation events.

Implemented by providers which want to provide information about or want to reposition contained HWND-based elements.

IRawElementProviderOverrideType

Implemented by providers which want to provide information about or want to reposition contained HWND-based elements.

IRawElementProviderSimple

UIAutomation provider interface, implemented by providers that want to expose properties for a single element. To expose properties and structure for more than a single element, see the derived **IRawElementProviderFragment** interface

IScrollInteropProvider

The Scroll pattern exposes a control's ability to change the portion of its visible region that is visible to the user by scrolling its content.

Examples: Listboxes

TreeViews other containers that maintain a content area larger than the control's visible region

Note that scrollbars themselves should not support the

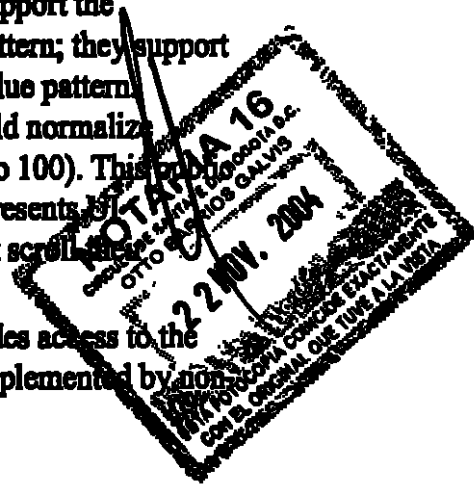
Scrollable pattern; they support the RangeValue pattern.

Servers should normalize scrolling (0 to 100). This

interface represents elements that scroll their content.

ISelectionInteropProvider

Provides access to the properties implemented by



(23

626

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ISelectionItemInteropProvider

"Longhorn" providers on containers that manage selection.

Provides access to methods and properties that define and manipulate selected items in a container. Is only supported on logical elements that are children of elements that supports ISelectionInteropProvider and is itself selectable.

ISortInteropProvider

Expose a container's current sort order and allow clients to programmatically re-sort its elements. Some containers maintain the sort order when inserting new items, or move their contents based on updated content (example: sorted Win32 listboxes). Other containers are only capable of performing a one-time sort that becomes out-of-date when new items are inserted (example: Excel).

ISplitInteropProvider

Exposes the ability to split an elements content area into multiple panes or interactive areas.

ITableInteropProvider

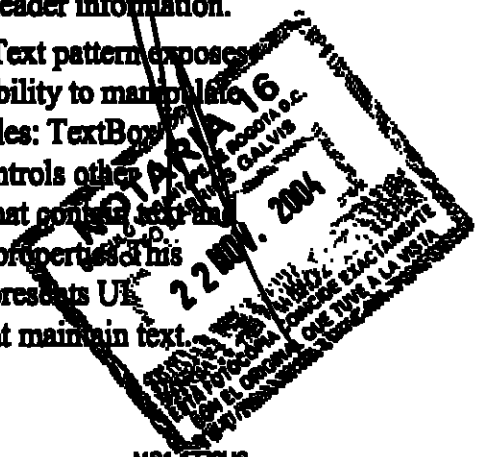
Identifies a grid that has header information.

ITableItemInteropProvider

Used to expose grid items with header information.

ITextInteropProvider

The Text pattern exposes a control's ability to manipulate text. Examples: TextBox, RichEdit controls other containers that contain text and text related properties. This interface represents UI elements that maintain text.



124

627

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ITextPointerInteropProvider

The Text pattern exposes a control's ability to manipulate text. Examples: TextBox RichEdit controls other containers that contain text and text related properties This interface represents UI elements that maintain text.

IValueInteropProvider

public interface that represents UI elements that are expressing a value

IVisualInformationInteropProvider

Used to express information about an image or an animation that is conveying information to the user.

IWindowInteropProvider

Expose an element's ability to change its on-screen position or size, as well as change the visual state and close it.

IZoomInteropProvider

Exposes the current zoom level in a control and allows clients to programmatically change it.

Enumerations

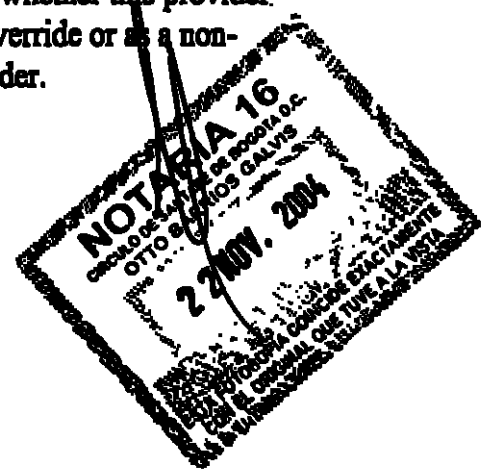
NavigateDirection

Directions for navigation the UIAutomation tree

RawElementProviderOverrideType is acting as an override or as a non-client-area provider.

Indicates whether this provider

Serialization Namespace



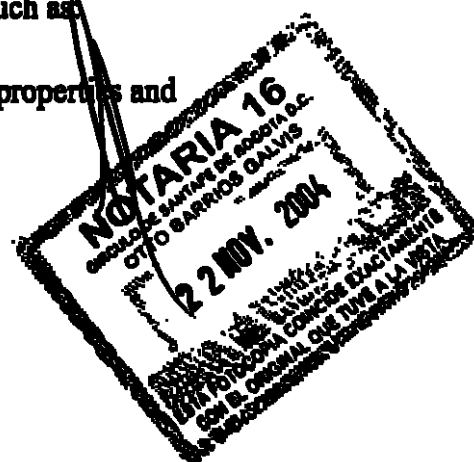
125

A serialization namespace 328 provides a parser that can load or save a hierarchy of objects (e.g., elements) from or to an XML file or a file with a binary representation. This process also sets properties associated with the objects and associates event handlers. Further, serialization namespace 328 provides functionality for serializing and de-serializing objects. Serialization is a process of taking a live, in-memory representation of objects, and producing a stream of data suitable for saving to a storage device (such as a disk) or sending across a network. Deserialization is the reverse process; i.e., taking a data stream and producing objects from the data stream. Deserialization is also referred to as "parsing", and the code that implements parsing is known as the parser.

Both serialization and deserialization support multiple streaming formats, such as an XML format referred to as XAML or a binary format referred to as BAML. When the parser is used with a markup compiler, it can generate "CAML", where the output is code to generate the serialized objects.

Serialization and deserialization support the full range of XAML features, including: standard syntax, compound properties, compact compound syntax, implicit child syntax, explicit children syntax, collection syntax with explicit collection tag, and collection syntax with implicit collection tag. Both serialization and deserialization are fully extensible. Classes can define how they wish to be serialized and deserialized through several mechanisms, such as:

- By default, class names are treated as markup tag names, and properties and events are treated as markup attributes of the same name.

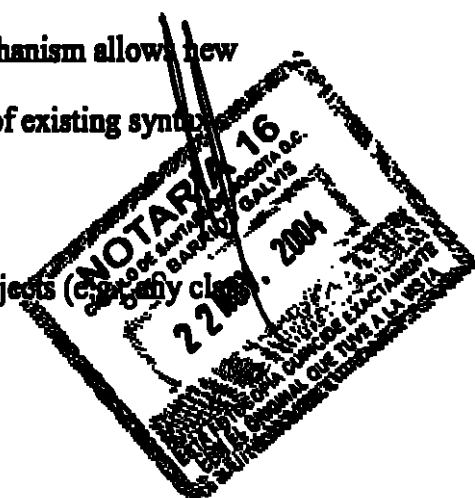


- Classes define type converters (auxiliary objects driving from `System.ComponentModel.TypeConverter`) to serialize and deserialize markup attribute values.
- Classes can support implicit children by implementing the `IAddChild` interface (parsing) and using CLR attributes to describe how to serialize implicit children.
- Classes can control whether a property can be serialized/deserialized by using the `DesignerSerializationVisibility`, and control whether it is necessary to serialize a property on a particular object by using the `DefaultValueAttribute` and `ShouldPersist` methods.
- Or, a component can serialize/deserialize itself using literal content, in which case the component's serialized form does not have to conform to the rules of XAML (the component needs to be well-formed XML).

Although the above examples use the word "markup", the same concepts apply to the binary format (BAML).

A second extensibility mechanism is "designer hooks", which allows another piece of code to plug into the serializer and deserializer, and insert and intercept attributes during parsing and serialization. This mechanism allows new markup features to be created, and allows custom processing of existing syntax. The markup compiler uses designer hooks to generate CAML.

Serialization and deserialization support a variety of objects (e.g., any class driving from `System.Object`). Classes driving from



670
630

System.Windows.DependencyObject can define dependency properties, which get two extra features:

- Serialization can automatically tell whether a property needs to be serialized (no need for DefaultValueAttribute/ShouldPersist)
- The parser can perform certain optimizations to load faster

Serialization and deserialization support XML namespaces and definition files that map XML namespaces to CLR namespaces. The Mapper class allows the definition of arbitrary mappings from XML namespaces and tag names into CLR classes.

The following tables list example members exposed by the System.Windows.Serialization namespace.

Classes

CustomXamlDeserializer

The base class for the deserialization helper. This should be subclassed if design or edit time deserialization direction of the parser is required.

CustomXamlSerializer

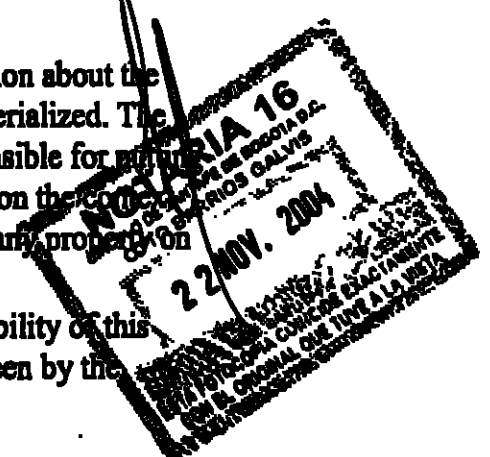
The base class for the serialization helper. This should be subclassed if design or edit time serialization direction of the serializer is required.

DependencyPropertyContext

Context information about the current property being serialized. The XamlSerializer is responsible for putting an instance of this class on the context stack before serializing any property on an object.

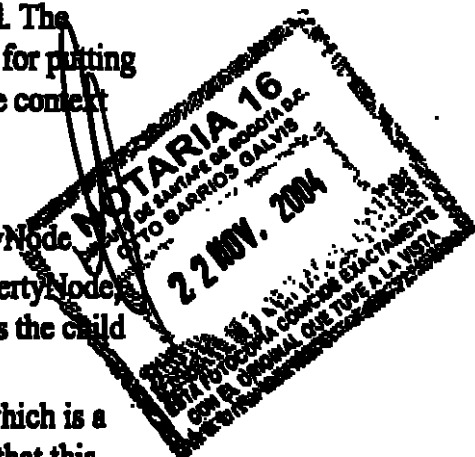
DPSerializationVisibilityAttribute

Specifies the visibility of this property or method as seen by the



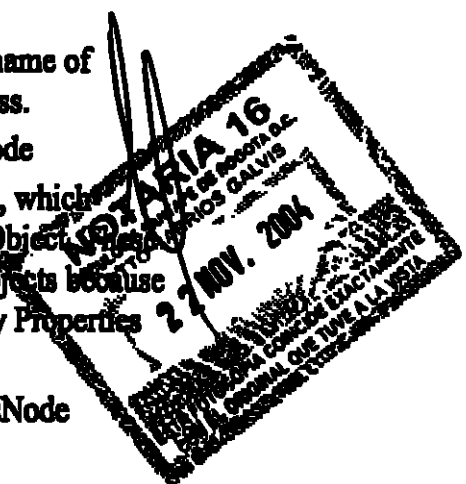
128

	designer serializer.
1	
2	Abstract base class for
	ObjectContext and PropertyContext.
3	Attribute for specifying the
	LayoutType for a tag.
4	Handles mapping between
	Extensible Markup Language (XML)
5	namespace Uniform Resource Identifier
	(URI) and Microsoft® .NET namespace
6	types.
7	Contains information the Mapper
	uses for Mapping between an xml
8	NamespaceUri and what Assembly,
	uriNamespace to look in.
9	Context information about the
	current object being serialized. The
10	XamlSerializer is responsible for putting
11	an instance of this class on the context
	stack in SerializeObject.
12	"Avalon" Element Parsing class
13	used to Create an "Avalon" Tree.
	Provides all the context
14	information required by Parser
15	Context information about the
	current property being serialized. The
16	XamlSerializer is responsible for putting
17	an instance of this class on the context
	stack before serializing any property on
18	an object.
	Context information about the
19	current object being serialized. The
20	XamlSerializer is responsible for putting
	an instance of this class on the context
21	stack in SerializeObject.
22	XamlAttributeNode
	XamlAttributeNode
23	XamlClrArrayPropertyNode
	XamlClrArrayPropertyNode
24	XamlClrComplexPropertyNode
	XamlClrComplexPropertyNode,
	which is an object specified as the child
	of a clr object
25	XamlClrEventNode
	XamlClrEventNode, which is a
	clr event on any object. Note that this

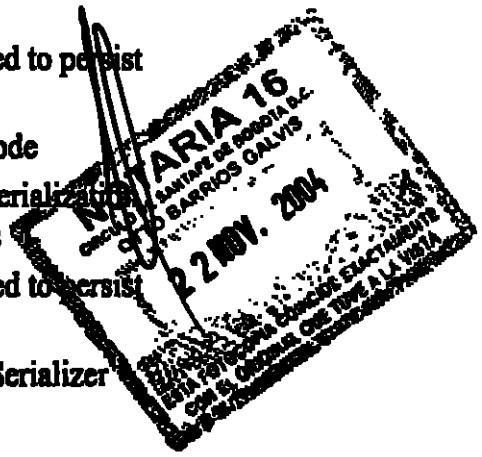


109

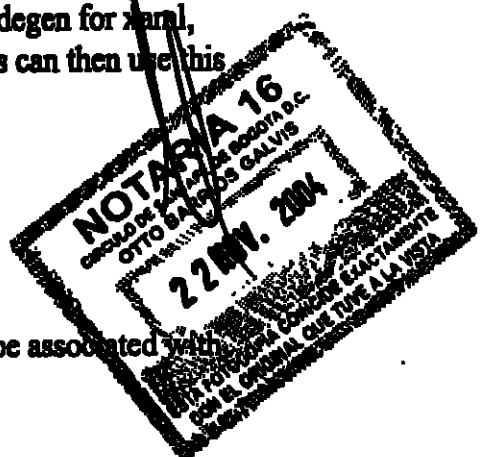
1		may be on a DependencyObject, or any other object type.
2	XamlClrObjectNode	XamlClrObjectNode, which is anything other than a DependencyObject
3		XamlClrPropertyNode, which is a
4	XamlClrPropertyNode	clr property on any object. Note that this may be on a DependencyObject, or any other object type.
5		Exception class for parser
6	XamlClrPropertyParseException	property parsing specific exceptions
7		XamlComplexDependencyPropertyNode, which is a DependencyProperty
8	XamlComplexDependencyPropertyNode	specified as the xml child of a DependencyObject
9		XamlComplexPropertySerializer
10	XamlComplexPropertySerializer	is used to serialize properties that can not be serialized as XML attributes. ComplexProperties are instead serialized as markup within the element.
11		XamlDefAttributeNode
12	XamlDefAttributeNode	XamlDefAttributeNode
13	XamlDefTagNode	XamlDefTagNode
14		Extends XamlSerializer to
15	XamlDependencyObjectSerializer	include animations (specific to Elements)
16		XamlDependencyPropertyNode,
17	XamlDependencyPropertyNode	which is a normal, simple DependencyProperty on a DependencyObject
18		Exception class for parser
19	XamlDependencyPropertyParseException	dynamic property parsing specific exceptions
20		Specifies the type name of
21	XamlDesignerSerializerAttribute	serializer for the given class.
22		XamlDocument Node
23	XamlElementNode	XamlElementNode, which represents a DependencyObject. These are different from CLR objects because they can have Dependency Properties associated with them.
24		XamlEndAttributesNode
25	XamlEndAttributesNode	XamlEndAttributesNode



1	XamlEndClrArrayPropertyNode	XamlEndClrArrayPropertyNode
2	XamlEndClrComplexPropertyNode	XamlEndClrComplexPropertyNode
3	de	de
4	XamlEndClrObjectNode	XamlEndClrObjectNode
5	XamlEndComplexDependencyPropertyNode	XamlEndComplexDependencyPropertyNode
6	XamlEndDocumentNode	XamlEndDocumentNode
7	XamlEndElementNode	XamlEndElementNode
8	XamlEndIDictionaryPropertyNode	XamlEndIDictionaryPropertyNode
9	e	e
10	XamlEndIListPropertyNode	XamlEndIListPropertyNode
11	XamlEndResourceNode	XamlEndResourceNode
12	XamlEndStartElementNode	XamlEndStartElementNode
13	XamlIDictionaryPropertyNode	XamlIDictionaryPropertyNode
14	XamlIEnumeratorSerializer	Serializes the node collection pointed to by the property.
15	XamlIListPropertyNode	XamlIListPropertyNode
16	XamlIListSerializer	Serializes the IList pointed to by the property.
17	XamlIncludeTagNode	XamlIncludeTagNode
18	XamlLanguageNode	XamlLanguageNode
19	XamlLiteralContentNode	XamlLiteralContentNode
20	XamlNode	Base Node in which all others derive
21	XamlParseException	Exception class for parser specific exceptions
22	XamlPIMappingNode	XamlPIMappingNode which maps an xml namespace to a clr namespace and assembly
23	XamlResourceNode	XamlResourceNode
24	XamlRootSerializer	XamlSerializer is used to persist logical tree.
25	XamlRoutedEventNode	XamlRoutedEventNode
26	XamlSerializationCallbackException	Exception class for serialization callback specific exceptions
27	ion	
28	XamlSerializer	XamlSerializer is used to persist the logical tree.
29	XamlSerializerBase	Base class for XamlSerializer



		providing common helper functions.
1	XamlTextNode	XamlTextNode
2	XamlTextNodeSerializer	Overrides XamlSerializer to special case TextNodes.
3	XamlXmlnsPropertyNode	XamlXmlnsPropertyNode
4	XmlAttributes	A class to encapsulate XML-specific attributes of a DependencyObject
5		
6	XmlnsDictionary	A dictionary to control XML namespace mappings
7	XmlParserDefaults	Public class used by Avalon.
8		
9		
10	Interfaces	
11	IAddChild	The IAddChild interface is used for parsing objects that allow objects or text underneath their tags in markup that do not map directly to a property.
12		
13	IBamlSerialize	Serialization interface for Dynamic Properties written to BAML files
14	ILoaded	This interface provides a read-only boolean property called IsLoaded, a clr event handler called Loaded, and DeferLoad and EndDeferLoad methods
15		
16	IPageConnector	Provides methods used internally by the BamlReader on compiled content.
17	IParseLiteralContent	For Literal Content - content which has its own parser and saver.
18		
19	IUriContext	The IUriContext interface allows elements (like Frame, PageViewer) and type converters (like ImageData TypeConverters) a way to ensure that base uri is set on them by the parser, codegen for xaml, baml and caml cases. The elements can then use this base uri to navigate.
20		
21		
22		
23	Enumerations	
24		
25	LayoutType	Layout Types that can be associated with an Object.



635

1		Describes the action the serializer or
2	SerializationAction	deserializer is to take after it has called back to the
3		CustomXamlSerializer or
4		CustomXamlDeserializer helpers
5		Describes the action the serializer or
6	SerializationErrorAction	deserializer is to take when an error has been
7		reported
8	XamlNodeType	Identifier for XamlNodes

7 Interop Namespace

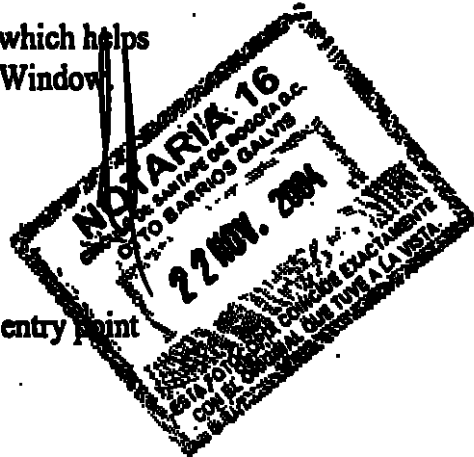
8 An interop namespace 330 provides a set of classes that enable
9 interoperability with other operating systems or computing platforms. The
10 following tables list example members exposed by the System.Windows.Interop
11 namespace.

12
13 Classes

14		MarshalByRefObject wrapper over
15	ApplicationProxy	Application class to allow interaction with
16		Application object across AppDomains and to
17		allow Application creation on a different thread.
18		Interop class used for implementing the
19	DocobjHost	managed part of a Docobj Server for browser
20		hosting
21	PresentationInteropHelper	VisualInteropHelper
22		Implements Avalon
23	WindowInteropHelper	WindowInteropHelper classes, which helps
24		interop b/w legacy and Avalon Window.

25 Delegates

26	AppEntryPoint	Delegate for the Application code's entry point
27		method.



Forms.Interop Namespace

A forms.interop namespace 332 provides an element that allows an application to host a form control operation.

System.IO.CompoundFile Namespace

Another namespace, System.IO.CompoundFile (not shown in Fig. 3) provides services to utilize a compound file in which various document distributable files are stored. These services allow for the encryption and compression of content. The services also support the storage of multiple renditions of the same content, such as a re-flowable document and a fixed-format document. The following tables list example members exposed by the System.IO.CompoundFile namespace.

Classes

- CertificateHelper**
- CompoundFileByteRangeReference**
- CompoundFileIndexReference**
- CompoundFileReference**
- CompoundFileReferenceCollection**

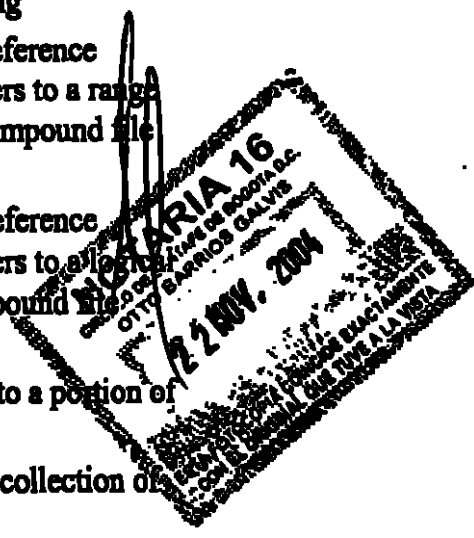
Helper to get a digital certificate for signing

Substream reference component that refers to a range of bytes within a compound file stream.

Substream reference component that refers to a logical entry within a compound file stream.

A reference to a portion of a compound file.

A read-only collection of



64X

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

CompoundFileStorageReference

compound file references.

Logical reference to a container storage

CompoundFileStreamReference

Logical reference to a container stream

CompressionTransform

CompressionTransform for use in Compound File DataSpaces

ContainerHelper

This class enables users to get access to the StorageRoot of the current container based on the loaded Application instance

DataSpaceManager

Used to manipulate the data spaces within a specific instance of the "Avalon" container. This is how data transform modules are plugged into the container to enable features like data compression and data encryption.

DigitalSignature

Read-only class that enables clients to inspect and validate existing digital signatures.

DigitalSignatureCollection

A read-only collection of digital signatures.

DigitalSignatureManager

This class is used to create, persist, and manipulate digital signatures in a compound file.

DocumentSummaryInfo

Contains property elements corresponding to the properties in the standard OLE document summary information property set.

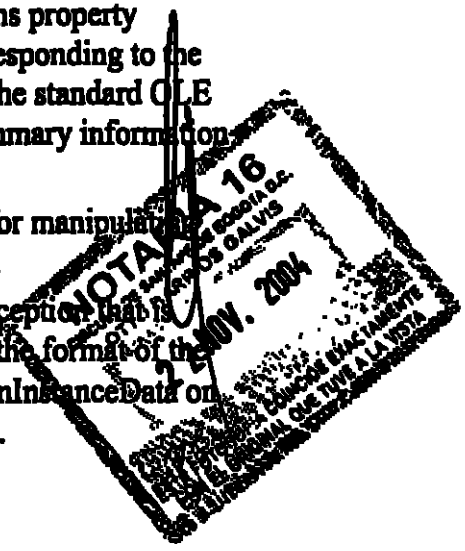
FormatVersion

Class for manipulating version object

InstanceDataFormatException

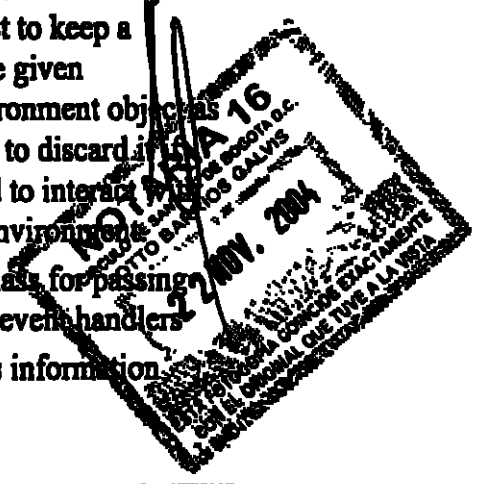
The exception that is thrown when the format of the DrmTransformInstanceData on disk is invalid.

OleProperty



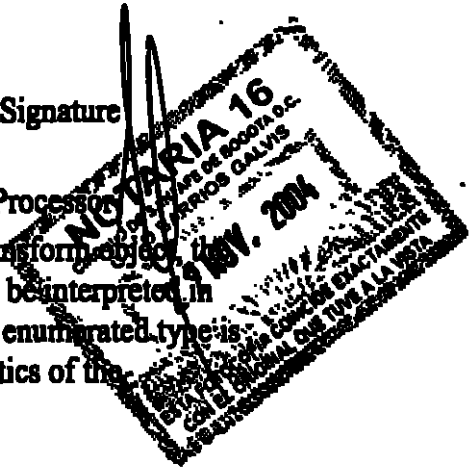
638

1	PropSet	
2		Class for manipulating
3	RenditionInfo	information of a rendition. This
4		class also provides APIs for
5	RenditionInfoCollection	adding and removing storages and
6		streams to and from the rendition.
7		A strongly typed collection
8	RenditionManager	of RenditionInfo objects
9		Class for manipulating
10		renditions in a container
11		Class implements
12	StorageInfo	IDataTransform interface, so it
13		can be used as a Container data
14	StorageRoot	transform that implements RM
15		encryption and decryption of the
16		content
17		Class for manipulating
18	StreamInfo	storages in the container file
19		Represents the main
20	SummaryInfo	container class. There is one
21		instance of the StorageRoot per
22		compound file.
23		Provides access for
24	TransformEnvironment	manipulating streams in the
25		container file.
		An instance of this class is
		given to each transform object as
		a means for the transform object
		to interact with the environment
		provided by the data space
		manager. It is not mandatory for a
		transform object to keep a
		reference on the given
		TransformEnvironment object as
		it might choose to discard it. If
		there is no need to interact with
		the transform environment
	TransformInitializationEventArgs	Public class for passing
		arguments into event handlers
	UseLicenseInfo	Contains information



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

	describing a use license returned from the Tungsten server.
UseLicenseInfoEnumerator	Represents an enumerator over the use licenses stored in the DRM Transform's instance data.
VersionTuple	Class for a version tuple which consists of major and minor numbers
XmlDigitalSignatureProcessor	Signature processor used by the digital signature manager to sign and validate data according to the W3C recommendation.
Interfaces	
IDataTransform	Interface to be implemented by all data transform objects
ILicenseStorage	This interface is used to decouple RMTransform and RMWizard by building custom implementation of this insterface we enable 3rd parties to take advantage of the RMWizard without forcing them to use RMTransform
ISignatureProcessor	Signature processor interface
IUnknownRCW	This interface is used to opaquely handle any COM interface (wrapped in the Runtime COM Wrapper) as an IUnknown.
Enumerations	
DigitalSignatureProcessor	Predefined Digital Signature Processors
SignatureProcessorNameType	Type of signature Processor
TransformIdentifierTypes	When naming a transform object, the string being passed in can be interpreted in one of several ways. This enumerated type is used to specify the semantics of the



identification string.

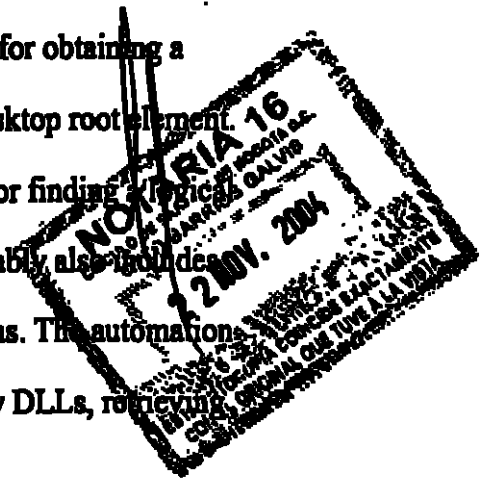
Delegates

InvalidSignatureHandler	Called with signatures that failed hash verification
TransformInitializeEventHandler	Delegate method for initializing transforms

System.Windows.Automation Namespace

A **System.Windows.Automation** namespace contains members used to support accessibility and user interface automation. The accessibility system includes a client side and a provider side. A system of tools includes client side automation tools including a client automation class for seeking user interface information. The client automation class includes event registration tools and logical element discovery tools. The set of tools further includes provider side automation tools for providing the client with user interface information. The provider side automation tools include an automation provider class having tools for providing the client with event information.

A client automation class provides UI automation methods for one or more clients. The client automation class contains methods that are not specific to any UI element. The client automation class may provide a method for obtaining a logical or raw element from a point, a window handle, or the desktop root element. The client automation class additionally may provide methods for finding element based input criteria. The client automation class preferably also includes a method for registering and un-registering for event notifications. The automation class preferably also provides helper functions for loading proxy DLLs, retrieving

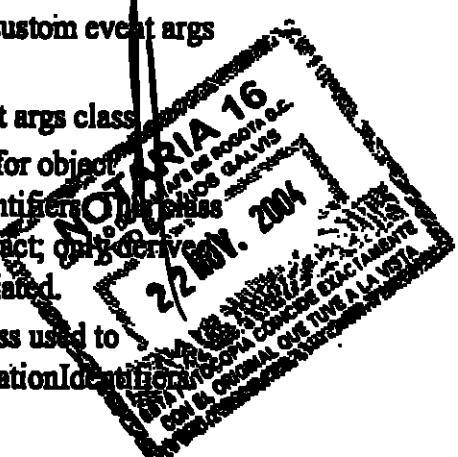


641

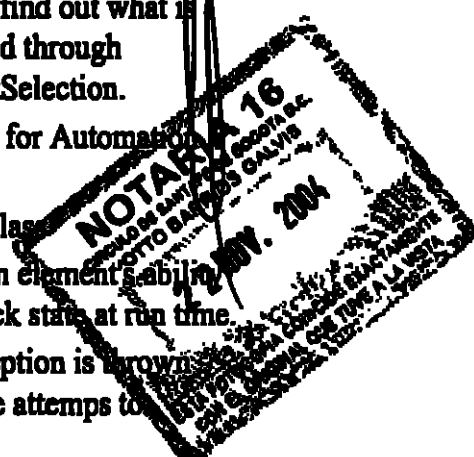
the localized names of properties and control patterns and performing element comparisons. The client automation class also includes methods for clients to listen for events. The following tables list example members exposed by the System.Windows.Automation namespace.

Classes

ApplicationWindowPattern	Exposes the behavior and information typically associated with a top-level application window. Clients can use this class to tile or cascade the application's multiple document interface (MDI) children, find its button on the taskbar, and locate well-known sections of its user interface such as toolbars and menus.
AssociatedInformationPattern	Exposes semantic and metadata for UI elements that represent other objects.
Automation	Contains User Interface (UI) Automation methods for clients (assistive technology or automated test script). These methods are not specific to a particular element.
AutomationEvent	Do not use. This class will be removed in future versions.
AutomationEventArgs	Pattern or custom event args class
AutomationFocusChangedEventArgs	Focus event args class
AutomationIdentifier	Base class for object identity-based identifiers. This class is effectively abstract; only derived classes are instantiated.
AutomationIdentifierProxy	Internal class used to deserialize AutomationIdentifier



1		Should not be used directly.
2	AutomationPattern	Do not use. This class will be removed in future versions.
3	AutomationPermission	Provides a set of permissions for accessing UI elements. This class cannot be inherited.
4		Provides methods and
5	AutomationPermissionAttribute	properties for AutomationPermissionAttribute. This class cannot be inherited.
6		Do not use. This class will be removed in future versions.
7	AutomationProperty	
8	AutomationPropertyChangedEventArgs	PropertyChanged event args class
9		Identifier for Automation
10	AutomationTextAttribute	Text Attributes
11		Represents the position of a character within text.
12	AutomationTextPointer	AutomationTextPointer provides methods and properties for accessing text and text navigation.
13		Used to get, set, add and remove selection.
14	AutomationTextRange	
15		Purpose: The AutomationTextSelection object handles all text selection management. The selection that the insertion pointer is on is the active selection. Example usages: It is used when clients want to add, remove, modify or set selection. Clients can also find out what is currently selected through AutomationTextSelection.
16	AutomationTextSelection	
17		Identifier for Automation
18	AutomationTextUnit	Text Units
19	BasePattern	Internal class
20	DockPattern	Expose an element's ability to change its dock state at run time.
21		This exception is thrown when client code attempts to
22	ElementNotEnabledException	
23		
24		
25		



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ElementPath

manipulate an element or control that is currently not enabled.

ElementPath provides criteria necessary to subsequently return to the logical element previously recorded, modified, or wholly created by the application provider.

ExpandCollapsePattern

Exposes a control's ability to expand to display more content or to collapse to hide content. Examples include menu buttons, Start button, tree-view item in Windows Explorer, and combo boxes.

GridItemPattern

Exposes the elements of a grid. Allows clients to quickly determine the coordinates of a grid item.

GridPattern

Exposes basic grid functionality, size, and navigation to specified cells.

HierarchyItemPattern

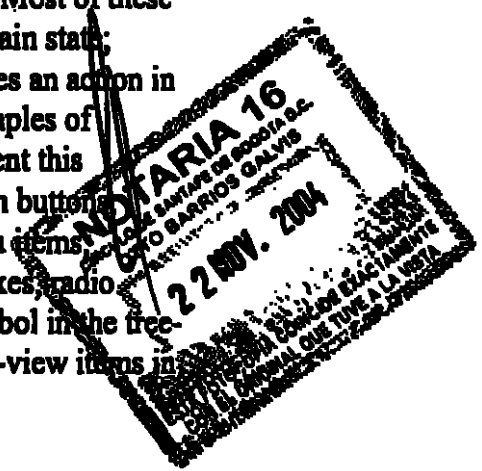
Exposes the hierarchical relationship between a control's user interface elements independent of their relationship in the logical tree.

Input

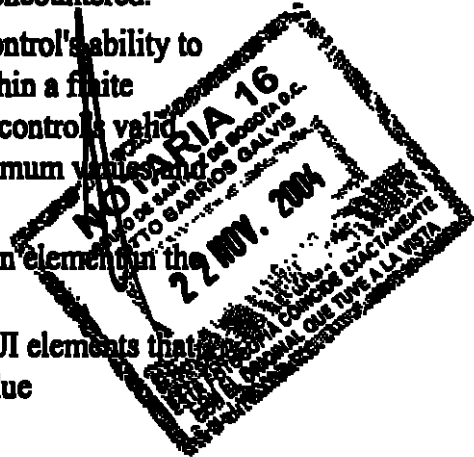
Provides methods for sending mouse and keyboard input

InvokePattern

Implemented by objects (controls) that perform a single, unambiguous action. Most of these controls do not maintain state; invoking them initiates an action in the application. Examples of controls that implement this interface include push buttons, toolbar buttons, menu items, hyperlinks, check boxes, radio buttons, the plus symbol in the tree-view control, and list-view items in Windows Explorer.



1		Represents a UI element in
2	LogicalElement	the logical tree regardless of
3		implementation ("Avalon",
4		Microsoft® Win32®).
5		Logical structure changed
6	LogicalStructureChangedEventArgs	event args class.
7		Exposes an elements's ability
8		to switch between multiple
9		representations of the same set of
10		information, data, or children.
11	MultipleViewPattern	Examples include ListView
12		(Thumbnails, Tiles, Icons, List,
13		Details) Excel charts (Pie, Line,
14		Bar, Cell Value with a formula),
15		and Outlook Calendar (Year,
16		Month, Week, Day).
17		The exception that is thrown
18		when the value of an argument is
19		outside of the allowable range of
20	NoClickablePointException	values, as defined by
21		GetClickablePoint. For example,
22		when the bounding rectangle is
23		empty, has no width or height or the
24		LogicalElement at that point is not
25		the same one that was called.
		This exception is thrown
		when there is a problem loading a
		proxy assembly. This can happen in
	ProxyAssemblyNotLoadedException	response to
		Automation.RegisterProxyAssembly
		or when loading the default
		proxies when the first hwnd base
		LogicalElement is encountered.
		Exposes a control's ability to
		manage a value within a finite
	RangeValuePattern	range. It conveys a control's valid
		minimum and maximum values and
		current value.
		Represents an element in the
	RawElement	raw element tree.
		Represents UI elements that
	ScrollPattern	are expressing a value



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

SelectedItemPattern

Represents individual items in containers that manage selection. UI Automation clients use this class to get information about, or to manipulate, UI elements that support the SelectionItemPattern control pattern. Examples include any items that can be selected in a control that supports the SelectionPattern control pattern, such as an item in a list box or tree view.

SelectionPattern

Represents containers that manage selection. Used by UI Automation clients to get information about, or to manipulate, user interface elements that support the SelectionPattern control pattern.

SortPattern

Exposes a container's current sort order and allows clients to programmatically re-sort elements.

SplitPattern

Represents windows that can clone themselves by creating an adjacent window.

TableItemPattern

Represents grid items with header information.

TablePattern

Represents a grid that has header information.

TextPattern

Represents text, such as an edit control.

TopLevelWindowEventArgs

TopLevelWindowEventArgs event args class

TreeLoadEventArgs

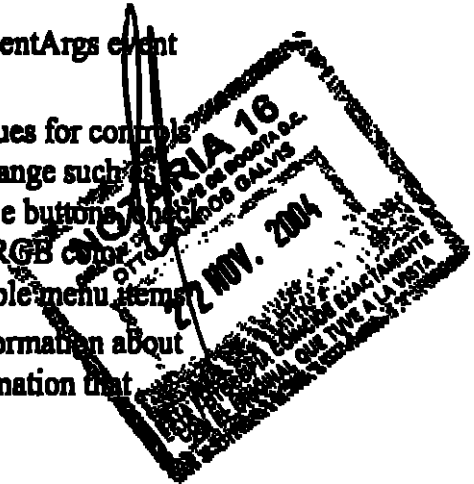
TreeLoadEventArgs event args class

ValuePattern

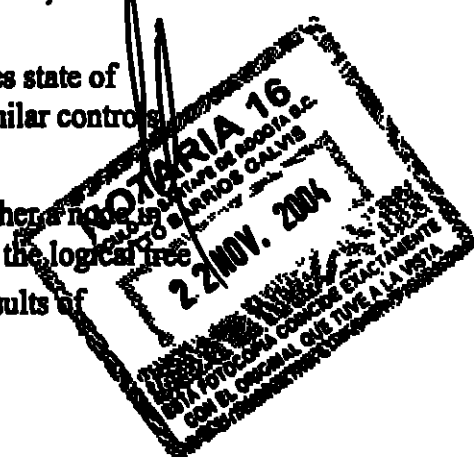
Exposes values for controls that do not span a range such as radio buttons, toggle buttons, checkboxes, edit boxes, RGB color values, and checkable menu items.

VisualInformationPattern

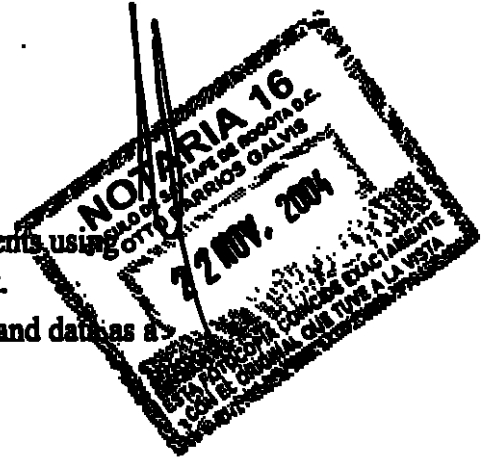
Exposes information about an image or an animation that



1		conveys information to the user
2		such as Internet Explorer's waving
3		flag or spinning globe that indicates
4		a document in downloading or
5		Windows Explorer's flying papers
6	WindowPattern	that indicate copying is in progress.
7		Exposes an element's ability
8		to change its on-screen position or
9		size. Examples include top-level
10		application windows (Word or
11		Windows Explorer), Outlook's
12		main window panes (folders, e-mail
13		messages, tasks), and MDI child
14		windows.
15	ZoomPattern	Exposes the current zoom
16		level in a control and allows clients
17		to change it programmatically.
18		
19	Enumerations	
20	AutomationPermissionFlag	Contains the access flags for an
21		AutomationPermission object.
22	ChildArrangement	Different ways children may be arranged
23	DockPosition	The edge of the container that the
24		dockable window will cling to.
25	ExpandCollapseState	Used by ExpandCollapse pattern to
26		indicate expanded/collapsed state
27	GetDataOptions	Options for GetData
28	HierarchyItemState	Enumeration that indicates the state of a
29		hierarchy item: collapsed, expanded, or
30		partially expanded
31	ItemCheckState	Enumeration that indicates state of
32		checkbox, radio buttons, and similar controls
33	LoadState	Tree state flags.
34	LogicalMapping	Values that indicate whether a node in
35		the raw tree will be displayed in the logical tree
36	ProxyFlags	Enum used to indicate results of
37		requesting a property



1	RowOrColumnMajor	Is the data data in this table best present by row or column
2	ScopeFlags	Flags used to define scope when listening to events
3	ScrollAmount	Used by ScrollPattern to indicate direction (forward or backward) and amount (page or line) by which to scroll
4	SendMouseInputFlags	Flags for SendMouseInput. These flags indicate whether movementnt took place, or whether buttons were pressed or released.
5	SplitStyles	Flag for the directions into which a window may split
6	StructureChangeType	Flags that indicate changes in the structure of the logical tree.
7	TextPointerGravity	Gravity determines the placement of a AutomationTextPointer when text is inserted at the AutomationTextPointer. Gravity is always from the text pointers point of view, i.e. I am before my character or I am after my character. When the user inserts a character at your text pointer location, Before means that your pointer will be placed before the new character; After means that the pointer will be placed After the new character.
8	WindowChangeType	Flags that indicate changes in top-level windows.
9	WindowInteractionState	The current state of the window for user interaction
10	WindowVisualState	States used to describe the visual state of the window. WindowVisualState follows the Office and HTML definition of WindowState.
11	Structures	
12	MatchCondition	Specifies criteria for finding elements using FindLogicalElement or FindRawElement.
13	NameAndData	Contains name, data as an object, and data as a string
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		



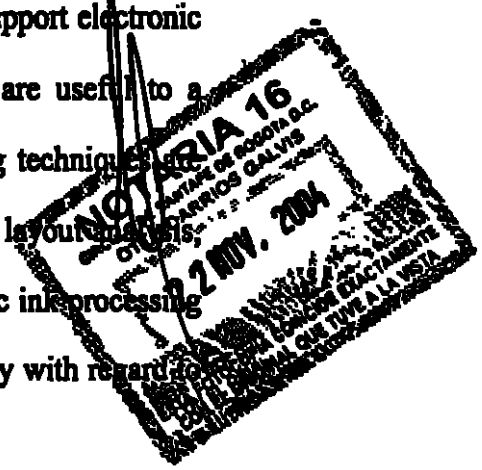
ProxyDescription Structure containing information about a proxy
SortInformation Information used to sort

Delegates

AutomationEventHandler	Delegate to handle AutomationEvents
AutomationFocusChangedEventHandler	Delegate to handle focus change events
AutomationPropertyChangedEventHandler	Delegate to handle Automation Property change events
LogicalStructureChangedEventHandler	Delegate to handle logical structure change events Implemented by HWND handlers; called by UI Automation framework to request a proxy for specified window. Should return a proxy if supported, or null if not supported.
ProxyFactoryCallback	
TopLevelWindowEventHandler	Delegate to handle top level window events

System.Windows.Ink Namespace

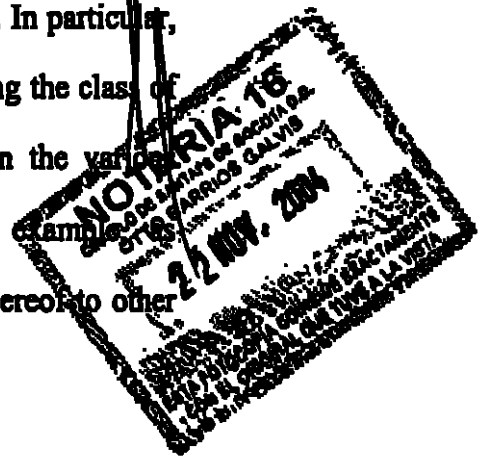
A System.Windows.Ink namespace provides classes that support electronic ink processing systems. Electronic ink processing techniques are useful to a variety of software applications. These electronic ink processing techniques are particularly applicable to the analysis of electronic ink, including layout analysis, classification, and recognition of electronic ink. Certain electronic ink processing techniques allow the electronic ink to be processed asynchronously with regard to



1 the operation of the software application implementing the techniques, so that the
2 electronic ink can be processed without stopping or significantly delaying the
3 operation of the software application. The software application can even continue
4 to accept new electronic ink input while previous electronic ink input is being
5 processed.

6 An application programming interface instantiates an ink analyzer object
7 that receives document data for a document containing electronic ink content from
8 a software application hosting the document and running on a first processing
9 thread. The ink analyzer object then employs the first thread to make a copy of
10 the document data, provides the copy of the document data to an electronic ink
11 analysis process, and returns control of the first processing thread to the analysis
12 process. After the analysis process has analyzed the electronic ink, the ink
13 analyzer object reconciles the results of the analysis process with current
14 document data for the document.

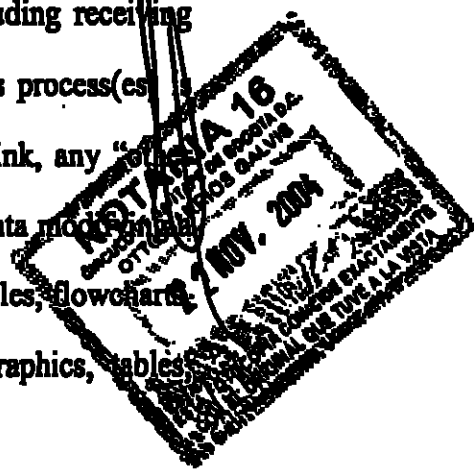
15 In particular embodiments, elements in a file or document may be described
16 based upon their spatial position relative to each other. For example, both an
17 electronic ink stroke and typewritten text may be described in terms of the same
18 spatial coordinate system. Using spatial information to describe the elements of a
19 document, the software application managing the document can maintain a data
20 structure describing the relationship between its document elements. In particular,
21 the software application can maintain a data structure both describing the class of
22 the various document elements and defining associations between the various
23 document elements. These associations can be defined, for example, to other
24 information used to link electronic ink stroke data or collections thereof to other
25



elements in the electronic document (such as words, lines, paragraphs, drawings, table cells, etc.).

By describing document elements in a file or document data structure based upon their spatial position, document elements for a variety of file types can employ common techniques for identifying and manipulating their document elements. More particularly, a variety of software applications can describe document elements within a document based upon their spatial position and employ this spatial position referencing to use common electronic ink analysis methods. Still further, by specifying a particular region of a document for analysis, each software application can limit the analysis process to only desired elements within a document.

To analyze new electronic ink input into a document, the software application managing the document modifies a data structure associated with the document to include the new ink to be analyzed. The software application then provides this data structure (or relevant portions thereof) to an ink analysis tool, which copies some or all of the data structure for analysis (and operates on this copy of the data that is independent of the application program's document data structure). The ink analysis tool passes the copy to an analysis process, such as a parsing process (e.g., a layout analysis process and/or a classification process). The software application may resume its normal operation, including receiving new electronic ink input and/or other data, while the ink analysis process(es) (are) being performed. In addition to receiving new electronic ink, any "data" may be received by the application program, for example, data modifying size, location, or content of existing ink, text, images, graphics, tables, flowcharts, diagrams, and the like; data adding additional text, images, graphics, tables,



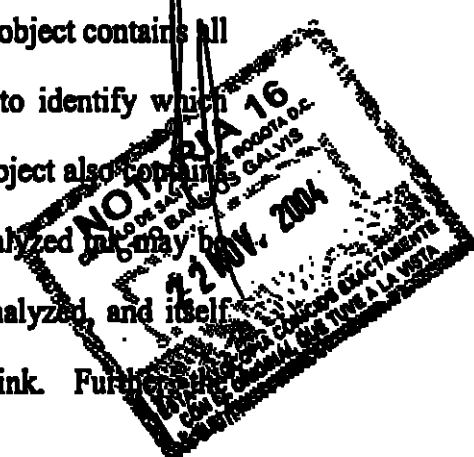
691

1 flowcharts, diagrams, and the like; data deleting existing text, images, graphics,
2 tables, flowcharts, diagrams, and the like. After all desired analysis processes are
3 completed, the analysis results are returned to the ink analysis tool.

4 An ink analysis tool performs a variety of functions to facilitate the
5 processing of electronic ink. The ink analysis tool can be implemented as a
6 programming interface, such as an API. Further, the ink analysis tool can be
7 implemented as a group of software object routines and related information that
8 can be called by a software application as needed to analyze ink in a document.

9 In one embodiment, an API embodying an implementation of an ink
10 analysis tool (referred hereafter to as an Ink Analysis API), may contain two core
11 classes. The first class is referred to as the "Analysis Context" class, and the
12 second class is the "Ink Analyzer" class. The components of the "Analysis
13 Context" class are used to create an analysis context object. The components of
14 the "Ink Analyzer" class are then used to create and employ an object that
15 provides a document independent analysis context object to an analysis process,
16 determine when the analysis results have been generated, and reconcile the
17 analysis results with the current state of a document.

18 The Analysis Context class is implemented by a host application to create
19 an analysis context object, which serves as a proxy view onto the internal
20 document tree of the software application. The analysis context object contains all
21 unanalyzed ink data, and the analysis context object is used to identify which
22 unanalyzed ink data should be analyzed. The analysis context object also contains
23 information about previously analyzed ink. This previously analyzed ink may be
24 used to decide how the currently unanalyzed ink should be analyzed, and itself
25 may be modified in the course of analyzing the unanalyzed ink. Further,



656

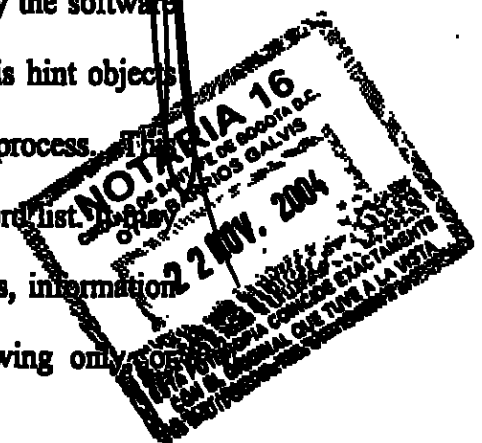
65L

1 analysis content object contains information about non-ink content of the
2 document, which is used to properly classify ink as annotations to the non-ink
3 content.

4 The Analysis Context class includes a constructor which, when called by
5 the software application, creates the analysis context object. This class may also
6 include various properties for the analysis context object, including the property
7 entitled "Dirty Region {get;}". The Dirty Region property defines the portion of
8 the document (and thus the portion of the analysis context object) that contains
9 unanalyzed ink data.

10 The Analysis Context class may also include a property entitled "Rootnode
11 {get;}", which identifies the topmost or root context node in the analysis context
12 object. This root context node contains, as child context nodes, all other context
13 nodes objects for the given analysis context object. In particular embodiments, the
14 root context node should be of the context node type "Root." In embodiments
15 where the application implements its own analysis context object, the analysis
16 context object may have other context nodes as siblings of the root context node,
17 but components of the Ink Analyzer class may be limited to considering context
18 nodes contained by the root context node.

19 The Analysis Context class may additionally include the property "Analysis
20 Hints {get;}", which returns an array of analysis hint objects set by the software
21 application. As will be discussed in more detail below, the analysis hint objects
22 may contain any type of information that may assist the analysis process. The
23 information may include, for examples, as factoids, guides, or a word list. The
24 also include information setting a language to be used for analysis, information
25 designating the unanalyzed ink as handwritten text only or drawing only.



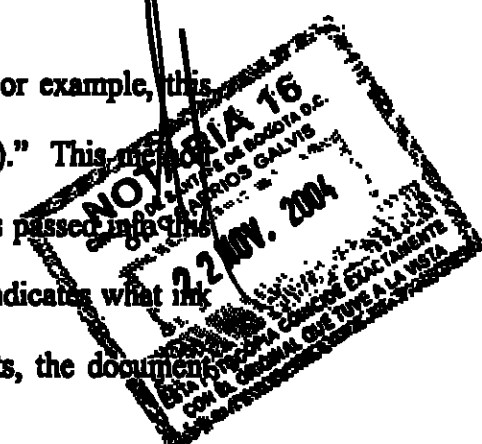
150

1 providing any kind of guidance to the parsing process, such as identifying the ink
2 as lists, tables, shapes, flow charts, connectors, containers and the like.

3 In addition to these properties, the Analysis Context class may also include
4 various methods that may be called by, e.g., the software application to have the
5 analysis context object execute a task. For example, the Analysis Context class
6 may include a method entitled "FindNode (Guid id)." Each node in the analysis
7 context object has a globally unique identifier (or GUID), and this method will
8 locate the node specified in the call anywhere in the analysis context object.

9 Like the Analysis Context class, the Ink Analyzer class also defines a
10 public constructor that allows the software application to create an instance of the
11 class (i.e., an ink analyzer object), along with various properties. For example, it
12 may contain a property entitled "User Interface Context {get; set;}," which defines
13 the processing thread to which the results of the analysis process are returned. This
14 property allows the results to be synchronized to another object. For example, if
15 this is set to the main form, parser results will be fired on the application main
16 thread. It may also contain the property "AnalysisOptions AnalysisOptions {get;
17 set;}", which specifies various criteria that may be used for the analysis process.
18 These criteria may include, for example, enabling text recognition, enabling the
19 use of tables, enabling the use of lists, enabling the use of annotations, and
20 enabling the use of connectors and containers.

21 The Ink Analyzer class will include various methods. For example, this
22 class may include a method entitled "AnalysisRegion Analyze ()" This method
23 starts a synchronous analysis process. Document element data is passed in this
24 method, which describes the current state of the document and indicates what ink
25 in the document needs to be analyzed. With some embodiments, the document

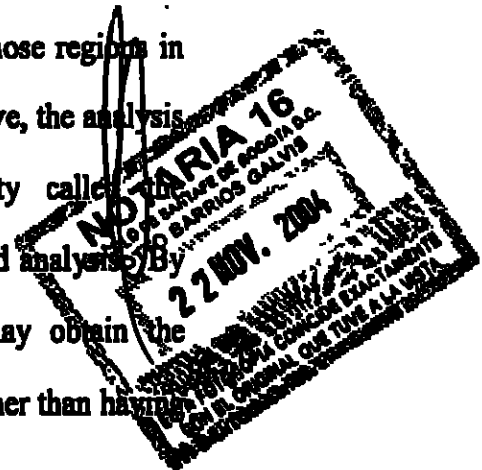


665
654

1 element data can be provided as the analysis context object (i.e., AnalysisRegion
2 Analyze (Analysis Context)), as noted above. Alternately, individual ink strokes
3 may be passed to the analysis process, either using a reference to the strokes (i.e.,
4 AnalysisRegion Analyze (Strokes)) or referenced as a property of the Ink
5 Analyzer object (e.g., InkAnalyzer.Strokes {get;set}) with no properties passed to
6 the Analyze method.

7 Once the analysis process is complete, this method will return a reference
8 to the document independent analysis context object that has been modified to
9 contain the results of the analysis process. The method also returns an
10 AnalysisRegion value describing the area in the document where results have been
11 calculated.

12 The Ink Analyzer class may also include a method entitled
13 "AnalysisRegion Analyze(AnalysisContext, waitRegion)." This method is the
14 same as the synchronous Analysis Region Analyze () method discussed above, but
15 it only has ink analyzed if results are needed in the specified waitRegion area.
16 More particularly, the call to this method will identify the analysis context object
17 for the document and the region of the analysis context object (referred to as the
18 "waitRegion") for which the analysis process should analyze synchronously. In
19 certain embodiments, all other regions of the analysis context object will be
20 ignored, unless the analysis process needs to analyze content in those regions in
21 order to complete its analysis of the waitRegion. As discussed above, the analysis
22 context object passed with this method contains a property called
23 "DirtyRegion" which describes the areas of the document that need analysis. By
24 specifying a particular waitRegion, the software application may obtain the
25 analysis results more quickly for one specific region of interest, rather than having



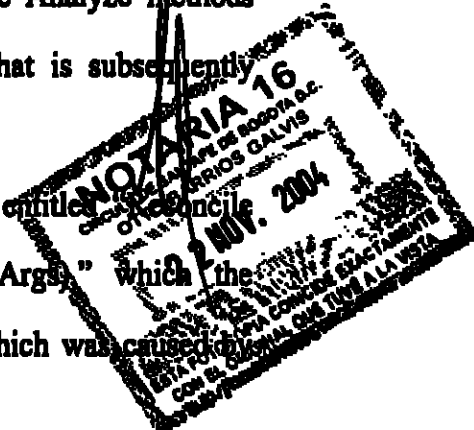
655

1 all of the ink data in the document analyzed. When either of these Analyze
2 methods is called, every available analysis process will be executed. Also,
3 because these Analyze methods are synchronous calls, there is no need to execute
4 the reconciliation process upon their conclusion, nor will an event fire once it is
5 complete.

6 The Ink Analyzer class may also include a method entitled
7 "BackgroundAnalyze(AnalysisContext)." This method starts the specified
8 analysis operation, but does so on a separate background analysis thread. Thus,
9 this method will return control to the main processing thread almost immediately
10 while the actual analysis operation completes in the background. In particular, this
11 method will return a value of "true" if the analysis process was successfully
12 started. Again, the AnalysisContext value passed in to the method identifies the
13 analysis context object for the document and indicates what ink in the document
14 needs to be analyzed. Once the analysis operation has completed on the
15 background thread, a Results event is raised to allow the software application to
16 access the results. The event contains the results and the reconcile method which is
17 used to incorporate the results back into the analysis context object for the current
18 state of the document when the results are returned.

19 Each of these three Analyze methods in turn call the method "Clone" in an
20 "Analysis Region" class. Using the "Clone" method, these Analyze methods
21 create the independent document analysis context object that is subsequently
22 modified by the analysis process to show the analysis results.

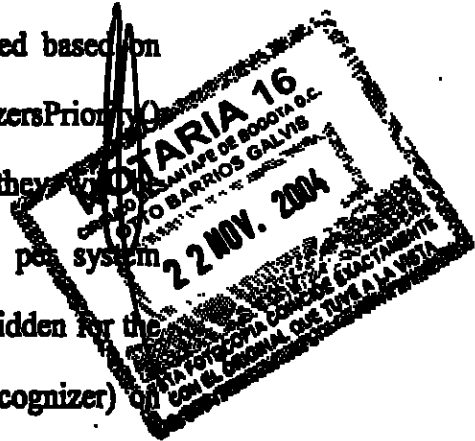
23 The Ink Analyzer class may also include a method entitled
24 (AnalysisContext current, AnalysisResultsEventArgs resultArgs) "which the
25 software application calls after receiving the results event which was raised by the



1 calling the BackgroundAnalyze(AnalysisContext) method. The Reconcile method
2 compares the analysis results contained in the document independent analysis
3 context object with the current version of the analysis context object maintained
4 by the software application. This method identifies the nodes that need to be
5 added and removed from the current version of the analysis context object, and
6 identifies if any of the following properties of an existing node has changed: its
7 recognition results, its location, the ink strokes associated with the node, or any
8 other data associated with the results of the analysis operation. This method also
9 writes these identified changes to the current version of the analysis context
10 object. This method is sensitive to the ordering of context node ordering, such as
11 the order of word context nodes on a line context node.

12 The analysis results (that is, the value of the property
13 AnalysisResultsEventArgs) are passed back with this method, as they contain a
14 public results structure and a private results structure. The public structure is
15 returned so the software application can preview the changes that will occur in the
16 reconcile stage. The private structure is included to prevent the software
17 application from changing any of the analysis results before the reconcile process.

18 The Ink Analyzer class may also include the methods entitled "Recognizers
19 RecognizersPriority()" and "SetHighestPriorityRecognizer(recognizer)." When
20 ink needs to be recognized, the appropriate recognizer will be used based on
21 language and capabilities. Accordingly, the Recognizers RecognizersPriority()
22 method returns the recognition processes in the order in which they will be
23 evaluated by the Ink Analyzer object. The order is determined per system
24 depending upon recognition processes are available, but can be overridden for the
25 software application by calling the SetHighestPriorityRecognizer(recognizer) of

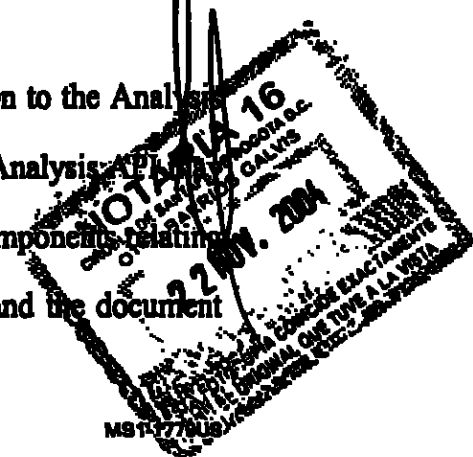


657

1 the Ink Analyzer object. The InkAnalyzer will enumerate through this ordered list
2 until an appropriate recognizer can be found. The
3 SetHighestPriorityRecognizer(recognizer) method raises the priority of a
4 recognition process. By raising the priority of a particular recognition process,
5 that recognition process will be used if it matches the needed language and
6 capabilities of the current recognition operation. In essence, the
7 SetHighestPriorityRecognizer(recognizer) pushes the designated recognition
8 process to the top of the list returned by the RecognizersPriority method.

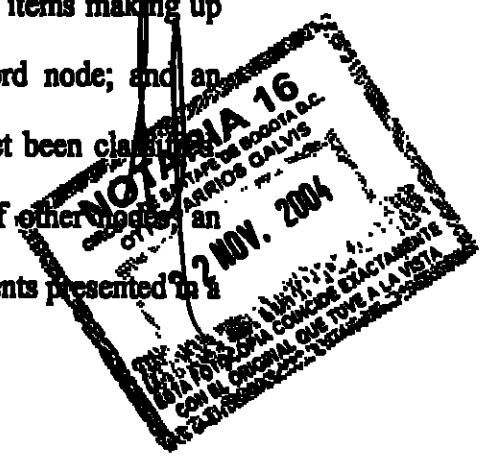
9 The Ink Analyzer class may also contain a method entitled
10 "AnalysisRegion Abort()," which may use an analysis context object as a
11 parameter. This method allows for a foreground or background analysis operation
12 to be terminated early. This method returns an analysis region that describes the
13 area that was being analyzed prior to the abort. Thus, if the software application
14 intends to continue the analysis operation at a later time, this region can be merged
15 into the DirtyRegion of the analysis context object for the current state of the
16 document. Still further, the Ink Analyzer class may include an event entitled
17 "AnalysisResultsEventHandler," which fires to the InkAnalyzer object as
18 frequently as practical. More particularly, this event may fire between analysis
19 processes and at least once every five seconds. This event can be used to provide
20 the application with an update as to the status of an ongoing asynchronous analysis
21 process (or processes).

22 The Ink Analysis API may also include classes in addition to the Analysis
23 Context class and the Ink Analyzer class. For example, the Ink Analysis API may
24 include a Context Node class. This class may include various components relating
25 to the context nodes that make up the analysis context object and the document



independent analysis context object, such as the property entitled "ContextNodeType Type {get;}." Each context node has a type, and there is a specific set of rules that each type should adhere to. This includes rules such as what types of child context nodes are allowed and whether or not strokes may be directly associated to the context node or only via its child context nodes.

The possible types of context nodes may be defined in a ContextNodeTypes enumeration and may include, for example, the following types: An InkAnnotation node, which represents ink data annotating non-text data; an InkDrawing node which represents ink data forming a drawing, an InkWord node, which represents ink data forming a word, a Line node, which contains one or more InkWord nodes and/or TextWord nodes for words forming a line of text; ListItem node, which may contain Paragraph, Image, or like nodes expected in a list; and a List node, which contains one or more ListItem nodes each describing an entry in the list. The node types may also include a NonInkDrawing node, representing a non-ink drawing image; an Object node, representing data not covered by other values of the ContextNodeType enumeration; a Paragraph node, which contains a one or more Line nodes corresponding to lines forming a paragraph; a Picture or Image node, representing a picture image; a Root node, which serves as the topmost node in an analysis context object; a Table node, which contains nodes representing items making up a table; a TextBox node, representing a text box; a TextWord node; and an UnclassifiedInk node, corresponding to ink data that has not yet been classified. The node types may also include a Group node, for groups of other nodes; an InkBullet node for bullet items, a Row node for documents elements presented in a

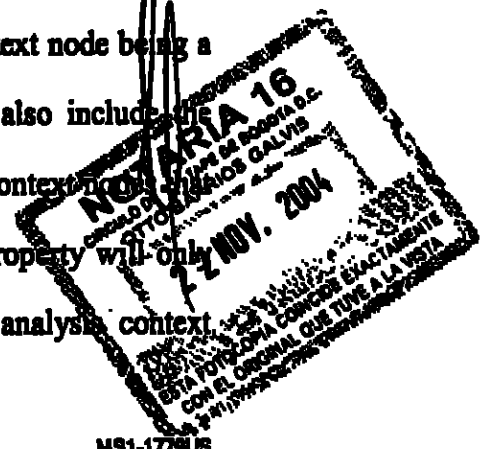


659

row of a table, and a Cell node for document elements presented in a cell of a table.

The Context Node class may also include the property entitled "GUID Id {get;}," which is a globally unique identifier for the current context node. In order to allow access to any desired context node, each context node within a single analysis context object should have unique identifier. This class may also include a property entitled "AnalysisRegion Location {get;}," which identifies the location in the document space where the relevant context node is actually positioned. An AnalysisRegion is a two-dimensional structure grouping one or more possibly disjoint rectangle like structures together. This class may also include the property entitled "StrokeCollection Strokes {get;}", which identifies the ink strokes associated with the relevant context node. With particular embodiments, only leaf context nodes (such as Word, Drawing and Bullet nodes) are allowed by the Ink Analysis API to have strokes. The software application may use this property to reference the strokes at the leaf node level by all ancestor context nodes (e.g., the root node would contain a strokes reference to all strokes in the relevant analysis context object.)

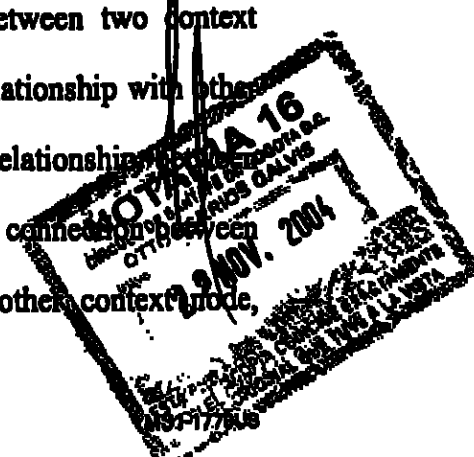
Further, this class may include the property entitled "ContextNode ParentNode {get;}," which identifies the parent context node containing the relevant context node. In particular embodiments; context nodes are always created to depend from a parent context node, with the Root context node being a static member of an analysis context object. This class may also include the property "ContextNode[] SubNodes {get;}" which identifies all context nodes which are direct children of the relevant context node. That is, this property will only identify those children context nodes down one level in the analysis context



object. For example, the value of this property for a Paragraph context node will only identify the line context nodes contained by the Paragraph node, and not the word context nodes that are children of the line context node.

This class may also include the property entitled "RecognitionResult
RecognitionResult {get;}", which provides the recognition result as calculated by the relevant recognition analysis process or processes, as the RecognitionResult can represent more than one line of text from in more than one language. The Recognition Result is available for every context node in the document independent analysis context object even though the RecognitionData property which is set by the recognition analysis process and is used to create the RecognitionResult object might only be set at one level of the context node tree to avoid duplication of data. If the node does not have a RecognitionData associated with it, it will either merge the recognition results of all of its subnodes or extract the recognition result from its parent. This class may also include the property entitled "Stream RecognitionData {get; set;}", which is the persistent form of the RecognitionResult value. Again, the recognition analysis process produces the Stream RecognitionData value that is set on the relevant context node. The RecognitionResult object is then constructed based on this value.

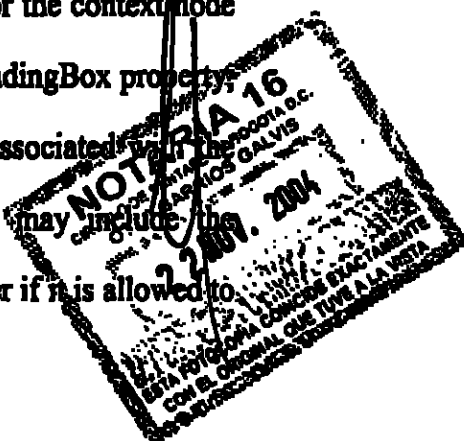
The Context Node class may further include a property entitled "ContextLink[] Links {get;}", which provides an array of ContextLink objects. A ContextLink object describes an alternative relationship between two context nodes. While context nodes typically have a parent-child relationship with other context nodes, a ContextLink allows for an alternative relationship between context nodes. For example, a ContextLink may allow for a connection between two context nodes, anchoring of one context node to another context node,



1 containment of one context node by another context node, or a desired type of link
2 defined by the software application. ContextLinks may be added to this array by
3 calling the AddLink method. Similarly, ContextLinks may be removed from this
4 array by calling the DeleteLink method.

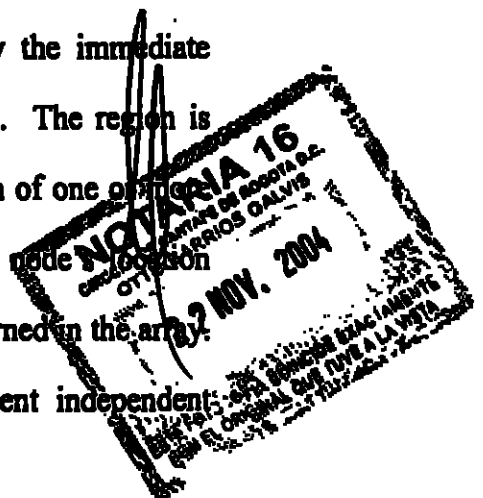
5 Still further, this class may include the properties "IsContainer {get;}" and
6 "IsInkLeaf {get;}." The property IsContainer {get;} has the value of "true" if the
7 relevant context node is not a leaf context node (that is, if the relevant context
8 node contains children context nodes and is thus considered a container context
9 node), and has the value of "false" otherwise. The IsInkLeaf {get;} property has
10 the value of "true" if the current context node is not a container context node, and
11 has a value of "false" otherwise. That is, if the current context node does not
12 contain any children context nodes, it is considered a leaf context node. In certain
13 embodiments, an InkLeaf context node is expected to contain references to stroke
14 data, whereas container context nodes do not have this restriction. Container
15 context nodes may or may not reference stroke data, as designated by the software
16 application.

17 The Context Node class may also contain the property "Rect
18 RotatedBoundingBox {get; set;}." The value of this property is calculated by a
19 layout and classification analysis process. If the ink data associated with the
20 relevant context node is written at an angle, then the bounds for the context node
21 will still be horizontally aligned. The value of the RotatedBoundingBox property,
22 however, will be aligned to the angle at which the ink data associated with the
23 relevant context node was written. Still further, this class may include the
24 property "ReClassifiable {get;}" which informs the InkAnalyzer if it is allowed to
25 modify the values of the relevant context node.



1 In addition to these properties, the Context Node class may also include
2 various methods. For example, this class may include a method entitled
3 "ContextNode CreateSubNode(ContextNodeType type)." This method allows the
4 creation of a child context node of a particular type. In one embodiment, this
5 method may only allow valid child types of the relevant context node to be
6 created, thereby preventing malformed data structures from being created. For
7 example, this method may only allow a Line context node to create InkWord and
8 TextWord child context nodes. This class may also contain a method entitled
9 "void DeleteSubNode(ContextNode node)," which deletes the referenced child
10 context node from the relevant analysis context object. However, in certain
11 embodiments, if the referenced context node still contains strokes or child context
12 nodes, then this method should fail. Also, if the reference context node is not a
13 direct child of the relevant context node, then this method should fail. If a
14 software application implements its own analysis context object and in turn
15 employs this method, it does not delete non-empty context nodes or context nodes
16 that are not direct children of the relevant context node to prevent malformed data
17 structures in the analysis context object.

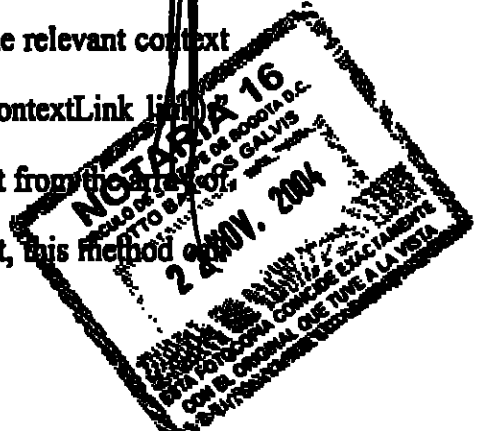
18 Additionally, this class may include the method "ContextNode[]
19 HitTestSubNodes(AnalysisRegion region)," which returns an array of context
20 node that are located in the specified region. However, only the immediate
21 children nodes of this element are returned, not all descendants. The region is
22 defined by the AnalysisRegion object, which may be a collection of one or more
23 rectangles. In particular embodiments, if any part of a context node
24 intersects the specified region, then that context node will be returned in the array.
25 This method is employed to, for example, create the document independent



1 analysis context object and to reconcile the analysis results with the analysis
2 context object corresponding to the current state of the document. Thus, this
3 method is frequently called, and should be optimized for fast repeated access by
4 the InkAnalyzer object.

5 The Context Node class may also contain a method entitled
6 "MoveStroke(Stroke stroke, ContextNode destination)." This method moves the
7 association of a stroke from one leaf context node to another. In certain
8 embodiments, this method is only used between leaf context nodes. It may also
9 include a method entitled "MoveSubNodeToPosition(int OldIndex, int
10 NewIndex)," which reorders the relevant context node with respect to its sibling
11 context nodes. For example, if the document has three words on a line, e.g., word
12 1, word 2 and word3, then their order is implied by the array of subnodes returned
13 from the parent context node. This method allows their order to be changed, so
14 that, relative to the relevant parent context node, word 1 is specified to be the last
15 word on the line by moving the context node for word 1 from position one to
16 position three.

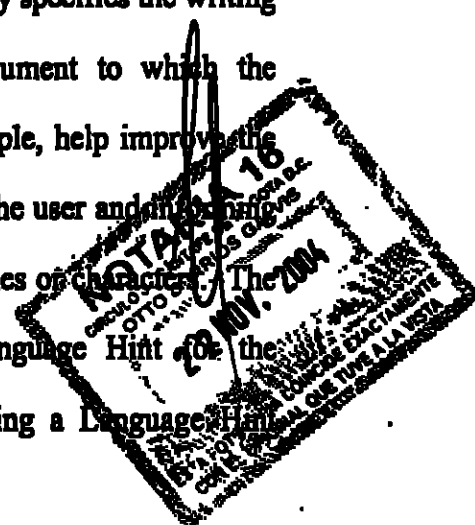
17 Still further, this class may include the method entitled
18 "AddLink(ContextLink link)," which adds a new ContextLink object to the
19 current context node. In particular embodiments, the ContextLink object should
20 contain a reference to the relevant context node in order for the ContextLink to be
21 successfully added to the array of ContextLinks associated to the relevant context
22 node. It may also contain the method entitled "DeleteLink(ContextLink link)".
23 This method deletes or removes the specified ContextLink object from the array of
24 ContextLinks for the relevant context node. In one embodiment, this method
25



always completes successfully, even if the ContextLink does not exist in the array of ContextLinks associated with the relevant context node.

The Ink Analysis API may also include an Analysis Hint class. As with many of the previously described classes, the Analysis Hint class may include a constructor, entitled "AnalysisHint()", which initializes an Analysis Hint object to an empty state. This class may also include a number of properties, including a property entitled "AnalysisRegion Location {get;}." This property specifies the location in the document (as an AnalysisRegion) to which the AnalysisHint is applicable. For example, if the document is a free form note with a title section at the top of the page, then the application could set an AnalysisHint for the title region to specify that a horizontal line of ink is expected in that region. This Analysis Hint will help to increase the accuracy of an analysis process.

This class may also include a property entitled "string Factoid {get; set;}," which specifies a particular "factoid" that is to be used for the location in the document to which the AnalysisHint is applicable. Factoids provide hints to a recognition process as to an expected use of ink data (e.g., as regular text, digits, postal codes, filenames, and web URLs). This class may also include the properties entitled "RecognizerGuide Guide {get; set;}" and "OverrideLanguageId {get; set;}." The RecognizerGuide Guide {get; set;} property specifies the writing guide that is to be applied to the location in the document to which the AnalysisHint is applicable. Writing guides may, for example, help improve the accuracy of a recognizer analysis process by specifying to the user and the recognizer the recognizer analysis process where the user will write lines of characters. The OverrideLanguageId {get; set;} property specifies a Language Hint for the document to which the AnalysisHint is applicable. Setting a Language Hint



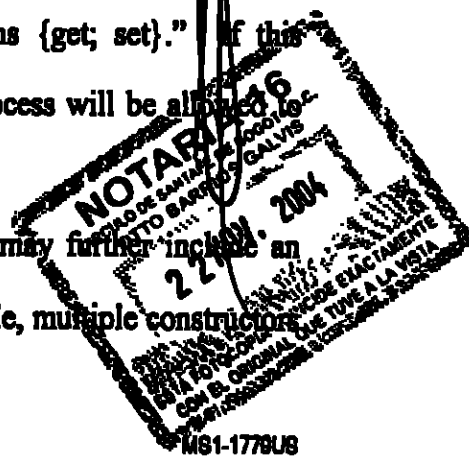
665

1 causes the InkAnalyzer object to use the designated language instead of the
2 language specified on the context node.

3 This class may also include a property entitled "PrefixText {get; set;},"
4 which specifies the text that is written or typed prior to a line of ink that is to be
5 recognized. Still further, this class may include a property entitled
6 "RecognitionModes RecognitionFlags {get; set;}," which specifies particular type
7 of modes a recognition process should respect at the location in the document to
8 which the AnalysisHint is applicable. Additionally, this class may include a
9 property entitled "SuffixText {get; set;}," which specifies the text that is written or
10 typed after to a line of ink that is to be recognized, and a property entitled
11 "WordList WordList {get; set;}," which specifies a particular set of words that
12 should be used by a recognition analysis process. Word lists may be used when
13 the expected recognition results are known before the user has actually written
14 input data, such as a list of medical terms that are expected to be written inside a
15 medical form.

16 Still further, this class may include a property entitled "WordMode {get;
17 set;}" If this value is "true," then the analysis process will bias itself to return a
18 single word for the entire analysis region. It may also include a property entitled
19 "Coerce {get; set;}," which, if "true," will force the analysis process to confine its
20 result to any factoid or wordlist value set in the relevant hint. This class may also
21 include a property entitled "AllowPartialDictionaryTerms {get; set;}" If this
22 property value is "true," then the recognition analysis process will be allowed to
23 return partial words from its recognition dictionary.

24 In particular embodiments, the Ink Analysis API may further include an
25 Analysis Region class. This class may include, for example, multiple constructors

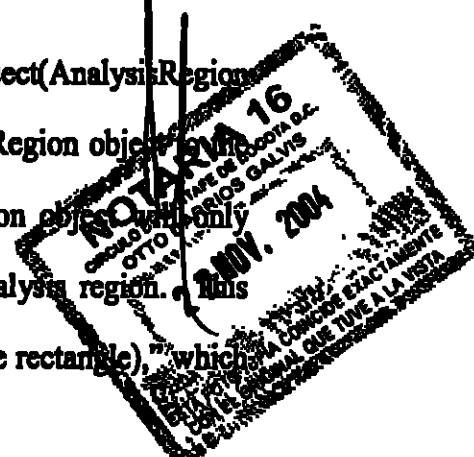


666

1 for constructing an AnalysisRegion object. For example, it may contain a first
 2 constructor for constructing an AnalysisRegion object having any region, a second
 3 constructor for constructing an AnalysisRegion object based upon the parameters
 4 for a two-dimensional rectangle, and a third constructor for constructing an
 5 AnalysisRegion object based upon four spatial coordinates. The default
 6 constructor may, for example, create an empty region. This class may also include
 7 a number of properties. For example, this class may include a property entitled
 8 "Rectangle Bounds {get;}," which retrieves the bounding rectangle for the
 9 AnalysisRegion, a property entitled "IsEmpty {get;}," which indicates whether the
 10 relevant AnalysisRegion object has an empty interior, and a property entitled
 11 "IsInfinite {get;}," which indicates whether the relevant AnalysisRegion is set to
 12 infinite or not.

13 This class may also include a number of methods, such as a method entitled
 14 "AnalysisRegion Clone()," which clones the relevant AnalysisRegion object. This
 15 class may also include a method entitled "Equals(AnalysisRegion otherRegion)," which
 16 tests whether the specified AnalysisRegion object (referred to as the
 17 otherRegion) is identical to the relevant AnalysisRegion object. This method
 18 returns a value of "true" if the interior of the specified Analysis Region object is
 19 identical to the interior of the relevant Analysis Region object, and otherwise
 20 returns a value of "false."

21 This class may further include a method "Intersect(AnalysisRegion
 22 regionToIntersect)," which crops down the relevant AnalysisRegion object to the
 23 specified analysis region. Thus, the resulting AnalysisRegion object will only
 24 include areas that overlapped or intersected the specified analysis region. This
 25 class may also include the method entitled "Intersect(Rectangle rectangle)," which



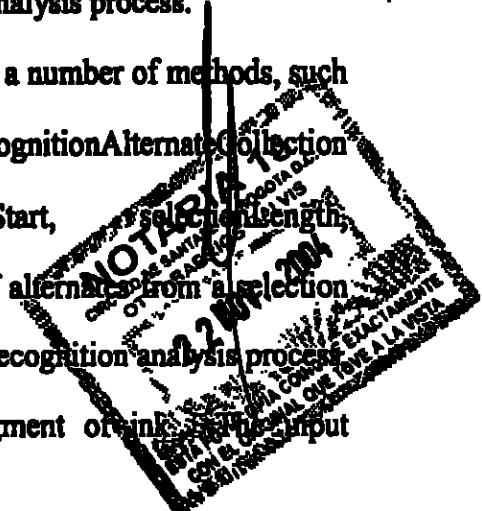
1 crops down the relevant AnalysisRegion object to the specified rectangle. Again,
2 the resulting AnalysisRegion object will only include areas that overlapped or
3 intersected the specified rectangle. It may also include the method entitled
4 "MakeEmpty()," which initializes the relevant AnalysisRegion object to an empty
5 interior, and the method entitled "MakeInfinite()," which sets the area occupied by
6 the relevant AnalysisRegion to be infinite. It may further include various methods
7 for uniting or separating differently defined areas, such as method entitled
8 "Union(AnalysisRegion regionToUnion)," which specifies an AnalysisRegion
9 object to union or add to the relevant AnalysisRegion object, and the method
10 entitled "Union(Rectangle rectangle)," which unions a specified rectangle to the
11 relevant Analysis Region object. With this method, the rectangle may be specified
12 in terms of the coordinate space for the relevant AnalysisRegion object. Of
13 course, this class may include numerous other methods for combining areas or
14 extracting one area from another based upon any desired definition for the areas.

15 The Ink Analysis API may also have a Recognition Result class. As with
16 many of the previously discussed classes, the Recognition Result class may
17 include one or more constructors. For example, this class may include a
18 constructor entitled "RecognitionResult(Stream lattice)", which constructs a
19 RecognitionResults object from a given recognition lattice. In particular
20 embodiments, a recognition lattice is a serialized form of the results from a
21 recognition process. This method may, for example, specify a recognition lattice
22 as a byte array to be used for the construction of the relevant RecognitionResult
23 object. It may also include a constructor entitled
24 "RecognitionResult(ContextNode node)," which constructs a RecognitionResults
25 object from a given context node. It may also include a constructor entitled

1 "RecognitionResult(string Text, int StrokeCount)," which constructs a
 2 RecognitionResults object from a specified text value, which in turn is associated
 3 to a specified number of strokes and might be used for correction if the recognition
 4 process did not come up with an alternate recognition value corresponding to the
 5 actual handwritten ink data. Still further, this class may include a constructor
 6 entitled "RecognitionResult(RognitionResult leftRecognitionResult,
 7 RecognitionResult rightRecognitionResult)," which constructs a
 8 RecognitionResults object by merging two existing Recognition Results objects
 9 together.

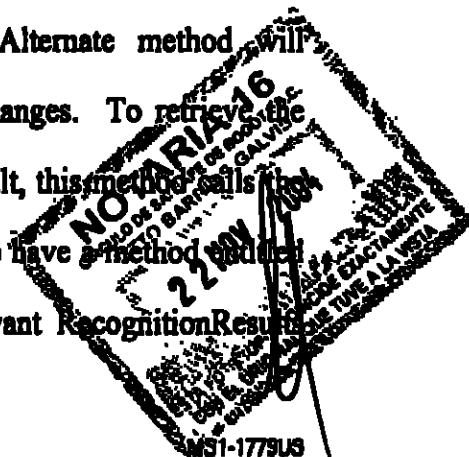
10 The Recognition Result class may also include one or more properties, such
 11 as a property entitled ""StrokeCollection StrokeCollection {get;}," which provides
 12 an array of stroke indexes representing a collection of strokes that are contained in
 13 a single ink object, and a property entitled "RecognitionAlternate TopAlternate
 14 {get;}," which provides the best alternate of the recognition result. This class may
 15 also include the property entitled "RecognitionConfidence RecognitionConfidence
 16 {get;}," which provides the level of confidence (e.g., strong, intermediate, or poor)
 17 of the top alternate selection for the current results from a recognition analysis
 18 process, and a property entitled "string TopString {get;}," which returns the best
 19 result string of the analysis results from a recognition analysis process.

20 The Recognition Results class may also include a number of methods, such
 21 as a method entitled "public RecognitionAlternateCollection
 22 GetAlternateCollectionFromSelection (selectionStart, selectionLength,
 23 maximumAlternates)," which specifies a collection of alternates from a selection
 24 within the best result string of analysis results from a recognition analysis process
 25 where each alternate corresponds to only one segment of ink in the input



parameters for this method may include, for example, a value which specifies the start of the text selection from which the collection of alternates is returned, a value that specifies the length of the text selection from which the collection of alternates is returned, and a value that specifies the maximum number of alternates to return. This method may then return the RecognitionAlternateCollection collection of alternates from a selection within the best result string of the recognition result.

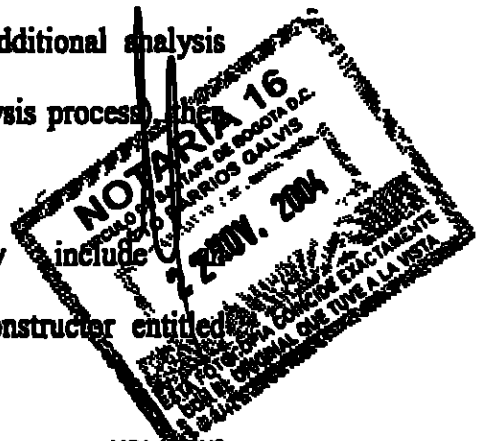
The Recognition Results class may further include a method entitled "RecognitionResult Merge(RecognitionResult left, string separator, RecognitionResult right)." This method may be used to create a new RecognitionResult object from a single string, resulting in a flat lattice, append a single string to the beginning or end of an existing RecognitionResult object, or concatenate a single string in between two existing RecognitionResult objects. This class may also include a method entitled "ModifyTopAlternate(RecognitionAlternate alternate)," which specifies the recognition result to be modified with a known alternate. With some embodiments, by default the best result string of the results of a recognition analysis process corresponds to the top alternate. However, this method can be used to specify that alternates other than the top alternate are used in the results of the recognition analysis process. If the new top alternate results in a different segmentation than the previous one, the ModifyTopAlternate method will automatically update the context nodes to reflect the changes. To retrieve the alternates that can be used to modify the recognition result, this method calls the GetAlternatesFromSelection method. This class may also have a method entitled "Stream Save()," which persistently maintains the relevant RecognitionResult



object in the form of a recognition lattice. A recognition lattice is a serialized format used to express the results from a recognition process.

The Ink Analysis API may also have an Analysis Options enumerated type. This type may contain one or more fields that specify how ink data will be analyzed by an analysis process, such a field entitled "const AnalysisOptions Default," which enables all available options for the analysis process. This field may, for example, enable text recognition, the use of tables, the use of lists, the use of annotations, the use of connectors and containers, and the use of intermediate results. This type may also include a field entitled "const AnalysisOptions EnableAnnotations," which enables and disables the detection of annotations, a field entitled "const AnalysisOptions EnableConnectorsAndContainers," which enables and disables the detection of connectors and containers, and a field entitled "const AnalysisOptions EnableIntermediateResults," enables and disables the return of analysis results to the software application between the use of different, sequential analysis processes (e.g., between a parsing process and a subsequent recognition process). This type may also have a field entitled "const AnalysisOptions EnableLists," which enables and disables the detection of lists, and a field entitled "const AnalysisOptions EnableTables," which enables and disables the detection of tables. This enumerated type may further include a field entitled "const AnalysisOptions EnableTextRecognition," which enables and disables a text recognition analysis process. However, if additional analysis processes are available (or different versions of the same analysis process), then this type may include additional AnalysisOptions accordingly.

Still further, the Ink Analysis API may include an AnalysisResultsEventArgs class. This class may have a constructor entitled

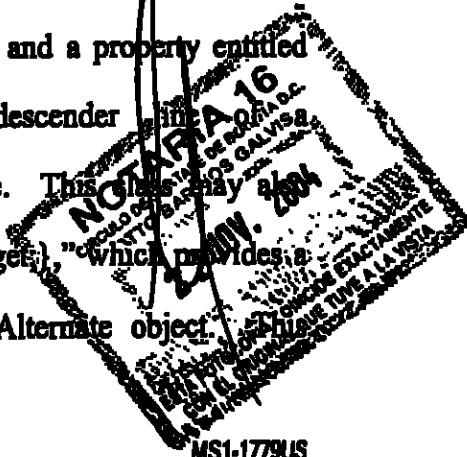


64

1 "public AnalysisResultsEventArgs()," which creates a data structure that contains
2 the analysis results and is returned to the software application when the
3 AnalysisResults event is raised. This class may also include a property entitled
4 "InkAnalyzer InkAnalyzer {get;}," which identifies the InkAnalyzer object that
5 performed the analysis process.

6 The API may also have a Line class, which may be useful with some types
7 of operating systems which recognize the use of a "Line" object representing a
8 geometric line. This class may include a constructor, such as a constructor entitled
9 "public Line(Point beginPoint, Point endPoint)," which creates a Line object. This
10 class may also include various properties, such as a property entitled "public Point
11 BeginPoint {get; set;}," which represents the beginning point of the line object
12 and a property entitled "public Point EndPoint {get; set;}," which represents the
13 ending point of the line object.

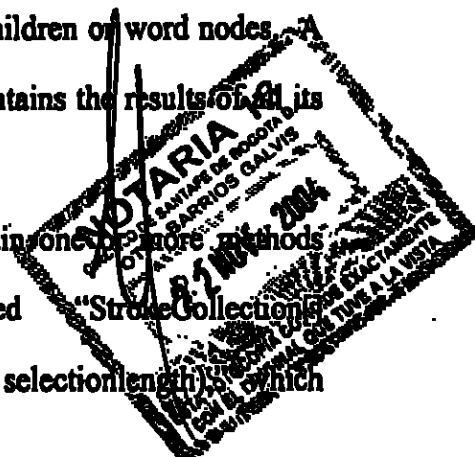
14 In addition to these classes, the Ink Analysis API may also contain a
15 Recognition Alternate class. This class may include elements representing the
16 possible word matches for segments of ink that are compared to a recognizer's
17 dictionary. For example, this class may include a property entitled "Line
18 Ascender {get;}," which provides the ascender line of a RecognitionAlternate
19 object that exists on a single line (with a line being represented as two points), a
20 property entitled "public Line Baseline {get;}," which provides the Baseline of a
21 RecognitionAlternate object that exists on a single line, and a property entitled
22 "Line Descender {get;}," which provides the descender line of a
23 RecognitionAlternate object that exists on a single line. This class may also
24 include a property entitled "RecognitionResult Extract {get;}," which provides a
25 RecognitionResults object for the current RecognitionAlternate object.



property can be used, for example, to extract the `RecognitionResult` object for a word from the `RecognitionResult` object for a line containing that word.

It may also include the property entitled "Line Midline {get;}," which provides the midline for a `RecognitionAlternate` object that exists on a single line, a property entitled "StrokeCollection Strokes {get;}," which provides the collection of strokes that are contained in an ink object (that is, it provides a `StrokeCollection` representing the strokes that are associated to the `RecognitionResult`), and a property entitled "StrokeCollection[] StrokesArray {get;}," which provides a collection of strokes that are contained in one or more ink objects, representing the strokes that are associated with the `RecognitionResult`. This class also may include a property entitled "RecognitionConfidence RecognitionConfidence {get;}," which provides the level of confidence (e.g., strong, intermediate, or poor) that a recognition analysis process has determined in the recognition of a `RecognitionAlternate` object or of a gesture. For non-line nodes, the lowest `RecognitionConfidence` of the children of the relevant context nodes will be returned. It may also contain the property entitled "string RecognizedString {get;}" which specifies the result string of the alternate. Thus, for any context node above a word context node, the results string is concatenated together by this method. For example, a line node will contain a results string that in turn contains the results of all its children or word nodes. A paragraph node will then contain a results string that contains the results of all its children or line nodes.

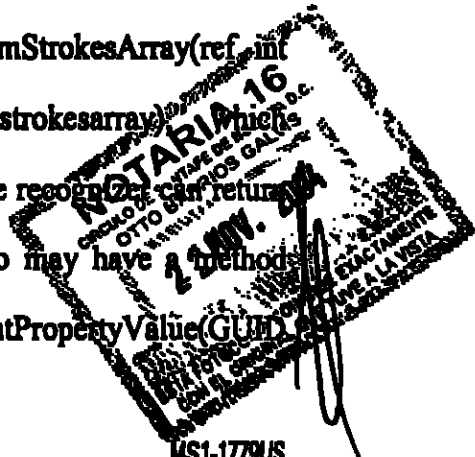
The `Recognition Alternate` class may also contain one or more methods including, for example, a method entitled "StrokeCollection GetStrokesArrayFromTextRange(int selectionstart, int selectionlength)" which



673

1 specifies a StrokeCollection from each ink object that corresponds to the known
 2 text range. This class may also contain a method entitled "StrokeCollection[]
 3 GetStrokesFromStrokesArrayRanges(StrokeCollection[] strokesArray)," which
 4 specifies the smallest collection of strokes that contains a known input collection
 5 of strokes and for which the recognizer can provide alternates. More particularly,
 6 the strokes are returned by an array of ink objects each containing an array of
 7 stroke indexes for the collection. The collection of ink strokes returned by this
 8 method may match the input collection, or it may be larger if the input collection
 9 matches only part of the smallest recognition result that includes all of the input
 10 strokes. This class may further include a method entitled "StrokeCollection
 11 GetStrokesFromStrokesRanges(StrokeCollection strokes)," which specifies the
 12 smallest collection of strokes that contains a known input collection of strokes and
 13 for which the recognizer can provide alternates, and a method entitled
 14 "StrokeCollection GetStrokesFromTextRange(int selectionstart, int
 15 selectionlength)," which specifies the StrokeCollection that corresponds to the
 16 known text range.

17 This class may further include a method entitled "void
 18 GetTextRangeFromStrokes(ref int selectionstart, ref int selectionend,
 19 StrokeCollection strokes)," which specifies the smallest range of recognized text
 20 for which the recognizer can return an alternate that contains a known set of
 21 strokes, and a method entitled "void GetTextRangeFromStrokesArray(ref int
 22 selectionstart, ref int selectionend, StrokeCollection[] strokesarray)," which
 23 specifies the smallest range of recognized text for which the recognizer can return
 24 an alternate that contains a known set of strokes. It also may have a method
 25 entitled "RecognitionAlternateCollection SplitWithConstantPropertyValue(GUI)"

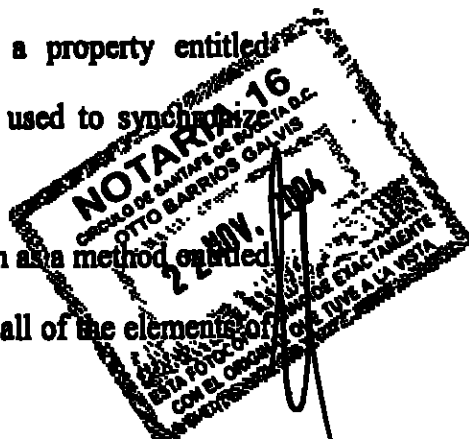


674

propertyType)," which returns a collection of alternates, which are a division of the alternate on which this method is called. Each alternate in the collection contains adjacent recognition segments which have the same property value for the property passed into the method. For example, this method can be used to obtain alternates that divide an original alternate by level of confidence boundaries (strong, intermediate, or poor) in the recognition result, line boundaries, or segment boundaries. It may further include a method entitled "byte[] GetPropertyValue(GUID propertyType)," which specifies the value of a known property of the alternate, such as the recognizer's confidence in the alternate. Not all recognition analysis processes will provide a value for all property types, however. Thus, this method provides the data for the types supported by the relevant recognition analysis process.

The Ink Analysis API may also include a Recognition Alternate Collection class. Like many of the classes discussed above, this class may include a constructor, entitled "RecognitionAlternateCollection()," for creating a RecognitionAlternateCollection object. This class may also include a number of properties, such as a property entitled "Count {get;}," which provides the number of objects or collections contained in a collection of alternate recognition values, a property entitled "IsSynchronized {get;}," which provides a value indicating whether access to the collection of alternate recognition values is synchronized with the software application (i.e., "thread safe"), and a property entitled "SyncRoot {get;}," which provides the object that can be used to synchronize access to the collection of alternate recognition values.

This class may also contain one or more methods, such as a method entitled "virtual void CopyTo(Array array, int index)," which copies all of the elements of

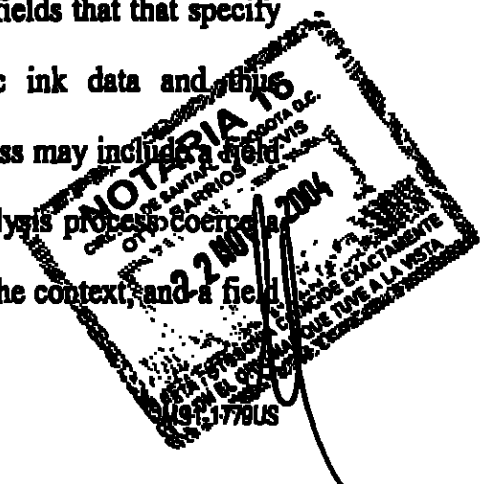


645

the current collection of alternate recognition values to the specified one-dimensional array, starting at the specified destination array index, and a method entitled "IEnumerator IEnumerable.GetEnumerator()," which is a standard implementation of IEnumerable that enables callers to use the for each construct to enumerate through each RecognitionAlternate in the collection of alternate recognition values. This class may also include a method entitled "RecognitionAlternateCollectionEnumerator GetEnumerator()," which returns a RecognitionAlternateCollectionEnumerator that contains all of the objects within the collection of recognition alternate values. This method may be used, for example, to retrieve each object in the collection of recognition alternate values.

The Ink Analysis API may additionally include a Recognition Confidence enumeration and a Recognition Mode enumeration, each of which may contain one or more fields relating to a recognition analysis process. For example, the Recognition Confidence class may contain multiple fields, such as a field entitled "Intermediate," indicating that the recognition analysis process is confident that the correct result is in the list of provided alternate recognition values, a field entitled "Poor," which indicates that the recognition analysis is not confident that the result is in the list of provided alternate recognition values, and a field entitled "Strong," which indicates that the recognition analysis process is confident that the best alternate in the alternate recognition values is correct.

Similarly, the Recognition Mode class may include fields that that specify how a recognition analysis process interprets electronic ink data and thus determines a recognition result string. For example, this class may include a field entitled "Coerce," which specifies that the recognition analysis process coerces a recognition result based on a factoid that was specified for the context, and a field

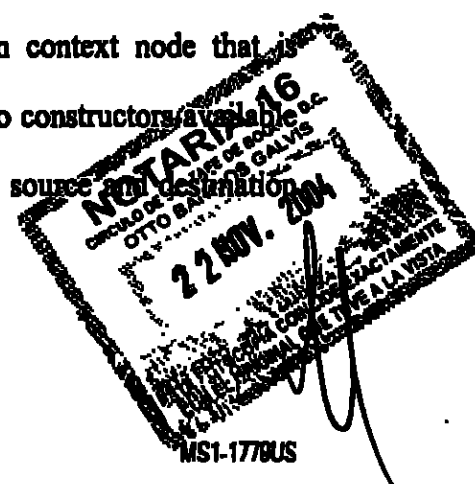


6x6

entitled "Line," which specifies that the recognition analysis process treat the electronic ink data as a single line. This class also may include a field entitled "None," which specifies that the recognition analysis process apply no recognition modes, and a field entitled "Segment," which specifies that the recognition analysis process treat the electronic ink data as forming a single word or character. Still further this class may include a field entitled "TopInkBreaksOnly," which disables multiple segmentation.

Still further, the Ink Analysis API may include a Context Link class, which defines how two context nodes may be linked together. The ContextLink object by itself represents which two context nodes are linked, the direction of the link, and the type of link. This class may include a property entitled "ContextNode SourceNode{get;}," which specifies the source context node that is being linked from another context node, a property entitled "ContextLinkType LinkType{get;}," which specifies the type of link relationship that exists between the source and destination context nodes, and a property entitled "CustomLinkType{get;}," which specifies that a custom link is being used. This situation would occur when an application decides to use the linking system of the Ink Analyzer API to represent application specific links beyond what the API can recognize.

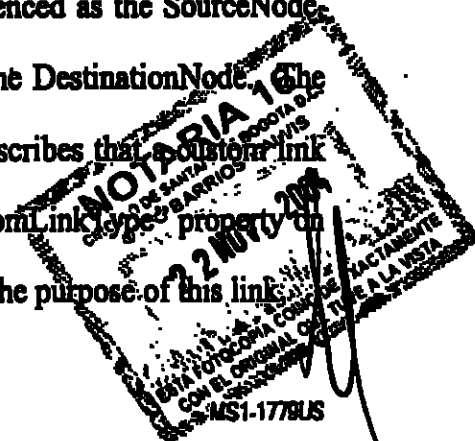
This class may also include a property entitled "ContextNode DestinationNode {get;}," which specifies the destination context node that is being linked from another context node. There may be two constructors available to this class, which create a relationship between existing source and destination context nodes.



647

1 This class may also include an enumeration entitled "ContextLinkType
2 enum," which defines the type of relationship shared by two context nodes. These
3 various link types may include, for example, an "AnchorsTo" type, which
4 describes that one node is anchored to the other node. Both nodes can use the
5 SourceNode or DestinationNode property based on the parsing situation. The link
6 types may also include the type "Contains," which describes that the one node
7 contains the other node. With this relationship, the container node could be
8 referenced as the SourceNode, while the containee node could be referenced as the
9 DestinationNode. The link types may further include a "PointsTo" type, which
10 describes that one node is pointing to another node. For this relationship, the node
11 doing the pointing could be referenced as the SourceNode, while the node being
12 pointed to could be referenced as the DestinationNode. Still further, the link types
13 may have a "PointsFrom" type, which describes that one node is pointing from the
14 other node. In this relationship, the node pointing away from the other node could
15 be referenced as the SourceNode, while the node being pointed from could be
16 referenced as the DestinationNode.

17 The link types may additionally include a "SpansHorizontally" type, which
18 describes that one node runs the length horizontally of another node, and a
19 "SpansVertically" type, which describes that one node runs the length vertically of
20 another node. For these types, the node covering (strike out, underline, margin
21 bar) the other node, usually written last, could be referenced as the SourceNode,
22 while the node being spanned could be referenced as the DestinationNode. The
23 link types may also include a "Custom" type, which describes that a custom link
24 type has been used. When this value is used, the "CustomLinkType" property on
25 the ContextLink object could provide more details as to the purpose of this link.

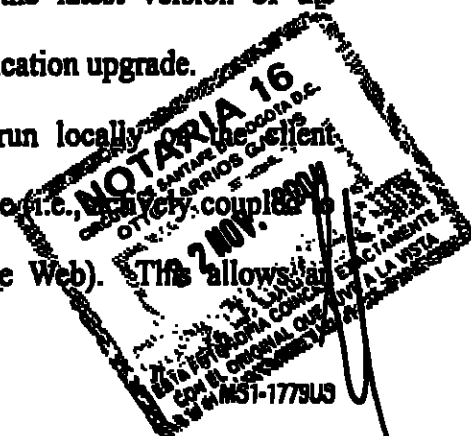


Application Model

A Windows Client integrates characteristics of the Web with characteristics of traditional desktop applications. The Application Model provides a framework for secure applications and simplifies development, deployment and maintenance of client applications. This framework provides a simple and consistent user experience. For example, local applications can leverage familiar browser-like features regardless of whether the application is hosted in the browser or is a standalone application, while retaining the benefits of executing on the local client. This framework allows users to leverage their familiarity with the Web, thereby increasing the user's comfort level and reducing the time required to learn to use a new application. The Application Model is part of the System.Windows namespace.

Applications utilizing the Application Model operate in a manner similar to Web pages. When a user browses to an application, the application is automatically installed without requiring user confirmation of the installation, rebooting the client system, or risking the malfunctioning of other applications. In one embodiment, applications download progressively, thereby providing an initial level of interactivity before the application is completely downloaded. Application updates are handled automatically and in a manner that is transparent to the user. Thus, the user always has access to the latest version of the application without having to explicitly perform an application upgrade.

Applications that use the Application Model run locally on the client system regardless of whether the client system is on-line (i.e., actively coupled to the Web) or off-line (i.e., not actively coupled to the Web). This allows an

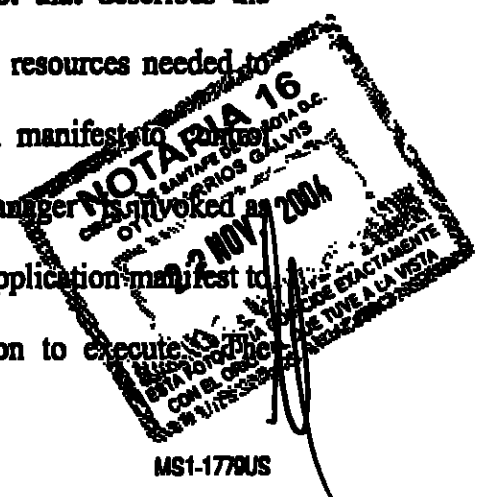


649

1 application to provide better performance than a server-based application that
2 needs an active Web connection and continually exchanges data with the server
3 across the Web. After an application is installed on a client system, the application
4 can be accessed from a "Start" menu (like a traditional desktop application) or by
5 navigating to the application (like a Web application). The Application Model
6 contains three primary parts: application lifecycle management, an application
7 framework and a navigation framework.

8 Two different types of applications are supported by the Application Model:
9 an "on-line application" and a "managed application". Applications utilizing the
10 Application Model can execute in the browser or in a standalone top-level
11 window. An "on-line application" is an application that executes from a server
12 and is hosted in a browser. The application can be cached for offline access or the
13 application may require certain on-line resources to execute properly. A "managed
14 application" is available off-line and is installed on the client. The operating
15 system services the managed application. An entry for the managed application
16 can be added to the "Start" menu on the client. Applications can be downloaded
17 progressively to allow the user to begin interacting with the application as it is
18 being downloaded rather than delaying interaction until an installation process has
19 finished.

20 Applications have an associated application manifest that describes the
21 application's dependencies, such as additional libraries and resources needed to
22 execute the application. An installer uses the application manifest to control
23 downloading and installation of the application. A "trust manager" is invoked as
24 part of the installation process. The trust manager uses the application manifest to
25 determine what permissions are needed for the application to execute.

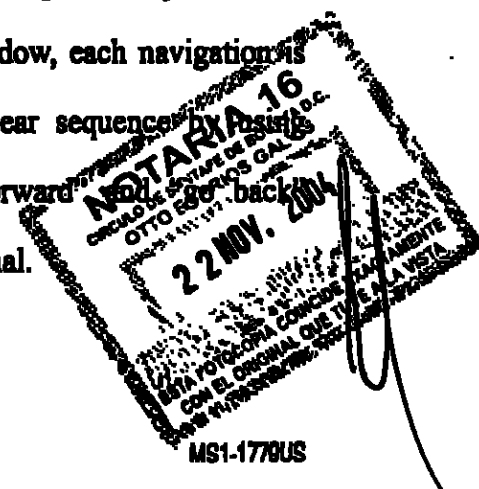


680

1 application manifest also specifies shell information, such as file associations and
2 whether to add an entry to the Start menu as well as the icon and text for the entry.

3 Applications utilizing the Application Model include markup, code,
4 resources and a manifest. An application is defined and scoped by its application
5 object, which is a global object that persists in memory for the lifetime of each
6 application session. The application object has knowledge of all the resources that
7 belong to the application and provides a boundary between itself and other
8 applications or external resources. The application framework uses the application
9 object to identify, reference and communicate with the application. The
10 application object is also used within the application to manage windows and
11 resources, specify startup and shutdown behavior, handle configuration settings,
12 specify visual styles for the application, share code, state and resources across
13 navigations, and handle application-wide events.

14 A navigation framework supports navigation-based applications that
15 leverage users' familiarity with navigation and journaling activities on the Web to
16 provide a more familiar, consistent user experience on the client system, regardless
17 of whether the application is hosted in the system browser or in a standalone top-
18 level window. Journaling is the process used by the navigation framework to track
19 navigation history. The journal allows users to retrace their steps backward and
20 forward in a linear navigation sequence. Whether a navigation experience is
21 hosted in the browser or in a standalone navigation window, each navigation is
22 persisted in the journal and can be revisited in a linear sequence by pressing
23 "forward" and "back" buttons or by invoking "go forward" and "go back"
24 methods. Each navigation window has an associated journal.



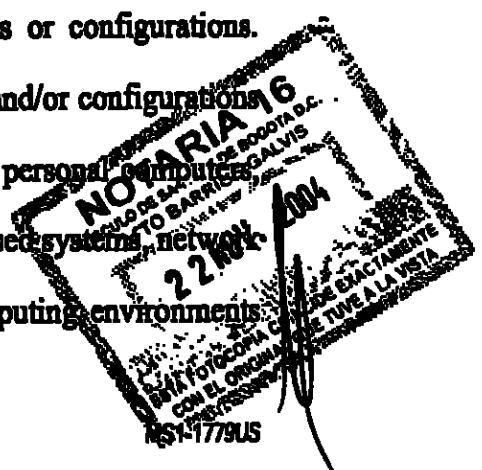
A NavigationApplication class simplifies the task of creating navigation-based applications by providing properties and events related to navigation. The NavigationApplication class includes a startup property that specifies the page or element to which the system navigates when the application is first launched. This class also has a properties collection that allows an application developer to share global state information across pages without having to subclass the application, and supports data binding to these properties.

EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

Fig. 4 illustrates an example of a suitable computing environment 400 within which the programming framework 132 may be implemented (either fully or partially). The computing environment 400 may be utilized in the computer and network architectures described herein.

The exemplary computing environment 400 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computing environment 400 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary computing environment 400.

The framework 132 may be implemented with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, multiprocessor systems, microprocessor-based systems, network PCs, minicomputers, mainframe computers, distributed computing environments,



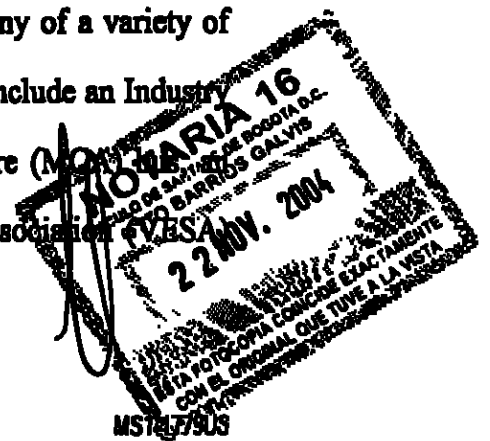
680
681

1 that include any of the above systems or devices, and so on. Compact or subset
2 versions of the framework may also be implemented in clients of limited
3 resources, such as cellular phones, personal digital assistants, handheld computers,
4 or other communication/computing devices.

5 The framework 132 may be described in the general context of computer-
6 executable instructions, such as program modules, being executed by one or more
7 computers or other devices. Generally, program modules include routines,
8 programs, objects, components, data structures, etc. that perform particular tasks
9 or implement particular abstract data types. The framework 132 may also be
10 practiced in distributed computing environments where tasks are performed by
11 remote processing devices that are linked through a communications network. In
12 a distributed computing environment, program modules may be located in both
13 local and remote computer storage media including memory storage devices.

14 The computing environment 400 includes a general-purpose computing
15 device in the form of a computer 402. The components of computer 402 can
16 include, by are not limited to, one or more processors or processing units 404, a
17 system memory 406, and a system bus 408 that couples various system
18 components including the processor 404 to the system memory 406.

19 The system bus 408 represents one or more of several possible types of bus
20 structures, including a memory bus or memory controller, a peripheral bus, an
21 accelerated graphics port, and a processor or local bus using any of a variety of
22 bus architectures. By way of example, such architectures can include an Industry
23 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus,
24 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) bus,
25



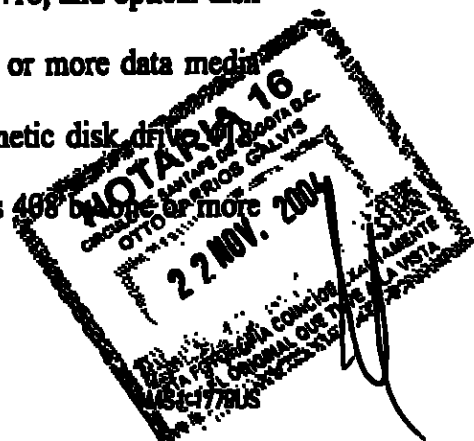
180

1 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
2 Mezzanine bus.

3 Computer 402 typically includes a variety of computer readable media.
4 Such media can be any available media that is accessible by computer 402 and
5 includes both volatile and non-volatile media, removable and non-removable
6 media.

7 The system memory 406 includes computer readable media in the form of
8 volatile memory, such as random access memory (RAM) 410, and/or non-volatile
9 memory, such as read only memory (ROM) 412. A basic input/output system
10 (BIOS) 414, containing the basic routines that help to transfer information
11 between elements within computer 402, such as during start-up, is stored in ROM
12 412. RAM 410 typically contains data and/or program modules that are
13 immediately accessible to and/or presently operated on by the processing unit 404.

14 Computer 402 may also include other removable/non-removable,
15 volatile/non-volatile computer storage media. By way of example, Fig. 4
16 illustrates a hard disk drive 416 for reading from and writing to a non-removable,
17 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading
18 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a "floppy
19 disk"), and an optical disk drive 422 for reading from and/or writing to a
20 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other
21 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk
22 drive 422 are each connected to the system bus 408 by one or more data media
23 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418
24 and optical disk drive 422 can be connected to the system bus 408 by one or more
25 interfaces (not shown).

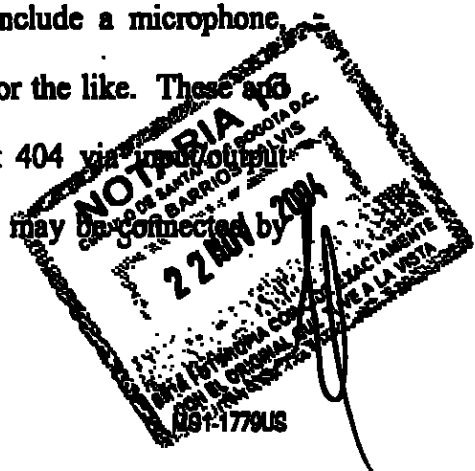


684

1 The disk drives and their associated computer-readable media provide non-
2 volatile storage of computer readable instructions, data structures, program
3 modules, and other data for computer 402. Although the example illustrates a hard
4 disk 416, a removable magnetic disk 420, and a removable optical disk 424, it is to
5 be appreciated that other types of computer readable media which can store data
6 that is accessible by a computer, such as magnetic cassettes or other magnetic
7 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
8 other optical storage, random access memories (RAM), read only memories
9 (ROM), electrically erasable programmable read-only memory (EEPROM), and
10 the like, can also be utilized to implement the exemplary computing system and
11 environment.

12 Any number of program modules can be stored on the hard disk 416,
13 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by
14 way of example, an operating system 426, one or more application programs 428,
15 other program modules 430, and program data 432. Each of the operating system
16 426, one or more application programs 428, other program modules 430, and
17 program data 432 (or some combination thereof) may include elements of the
18 programming framework 132.

19 A user can enter commands and information into computer 402 via input
20 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").
21 Other input devices 438 (not shown specifically) may include a microphone,
22 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
23 other input devices are connected to the processing unit 404 via input/output
24 interfaces 440 that are coupled to the system bus 408, but may be connected by other
25



686

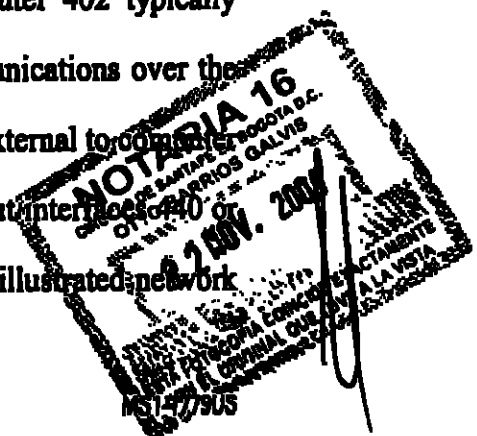
1 other interface and bus structures, such as a parallel port, game port, or a universal
2 serial bus (USB).

3 A monitor 442 or other type of display device can also be connected to the
4 system bus 408 via an interface, such as a video adapter 444. In addition to the
5 monitor 442, other output peripheral devices can include components such as
6 speakers (not shown) and a printer 446 which can be connected to computer 402
7 via the input/output interfaces 440.

8 Computer 402 can operate in a networked environment using logical
9 connections to one or more remote computers, such as a remote computing device
10 448. By way of example, the remote computing device 448 can be a personal
11 computer, portable computer, a server, a router, a network computer, a peer device
12 or other common network node, and so on. The remote computing device 448 is
13 illustrated as a portable computer that can include many or all of the elements and
14 features described herein relative to computer 402.

15 Logical connections between computer 402 and the remote computer 448
16 are depicted as a local area network (LAN) 450 and a general wide area network
17 (WAN) 452. Such networking environments are commonplace in offices,
18 enterprise-wide computer networks, intranets, and the Internet.

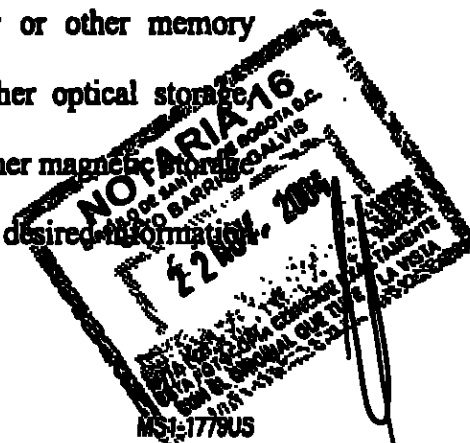
19 When implemented in a LAN networking environment, the computer 402 is
20 connected to a local network 450 via a network interface or adapter 454. When
21 implemented in a WAN networking environment, the computer 402 typically
22 includes a modem 456 or other means for establishing communications over the
23 wide network 452. The modem 456, which can be internal or external to computer
24 402, can be connected to the system bus 408 via the input/output interfaces 440 or
25 other appropriate mechanisms. It is to be appreciated that the illustrated network



1 connections are exemplary and that other means of establishing communication
2 link(s) between the computers 402 and 448 can be employed.

3 In a networked environment, such as that illustrated with computing
4 environment 400, program modules depicted relative to the computer 402, or
5 portions thereof, may be stored in a remote memory storage device. By way of
6 example, remote application programs 458 reside on a memory device of remote
7 computer 448. For purposes of illustration, application programs and other
8 executable program components such as the operating system are illustrated herein
9 as discrete blocks, although it is recognized that such programs and components
10 reside at various times in different storage components of the computing device
11 402, and are executed by the data processor(s) of the computer.

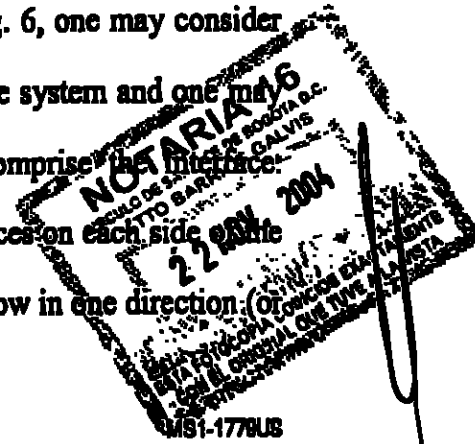
12 An implementation of the framework 132, and particularly, the API 142 or
13 calls made to the API 142, may be stored on or transmitted across some form of
14 computer readable media. Computer readable media can be any available media
15 that can be accessed by a computer. By way of example, and not limitation,
16 computer readable media may comprise "computer storage media" and
17 "communications media." "Computer storage media" include volatile and non-
18 volatile, removable and non-removable media implemented in any method or
19 technology for storage of information such as computer readable instructions, data
20 structures, program modules, or other data. Computer storage media includes, but
21 is not limited to, RAM, ROM, EEPROM, flash memory or other memory
22 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,
23 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage
24 devices, or any other medium which can be used to store the desired information
25 and which can be accessed by a computer.



1 "Communication media" typically embodies computer readable
 2 instructions, data structures, program modules, or other data in a modulated data
 3 signal, such as carrier wave or other transport mechanism. Communication media
 4 also includes any information delivery media. The term "modulated data signal"
 5 means a signal that has one or more of its characteristics set or changed in such a
 6 manner as to encode information in the signal. By way of example, and not
 7 limitation, communication media includes wired media such as a wired network or
 8 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
 9 other wireless media. Combinations of any of the above are also included within
 10 the scope of computer readable media.

11 Alternatively, portions of the framework may be implemented in hardware
 12 or a combination of hardware, software, and/or firmware. For example, one or
 13 more application specific integrated circuits (ASICs) or programmable logic
 14 devices (PLDs) could be designed or programmed to implement one or more
 15 portions of the framework.

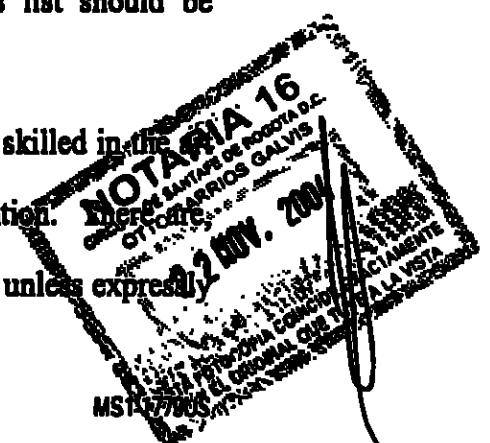
16 Notionally, a programming interface may be viewed generically, as shown
 17 in Fig. 5 or Fig. 6. Fig. 5 illustrates an interface Interface1 as a conduit through
 18 which first and second code segments communicate. Fig. 6 illustrates an interface
 19 as comprising interface objects I1 and I2 (which may or may not be part of the
 20 first and second code segments), which enable first and second code segments of a
 21 system to communicate via medium M. In the view of Fig. 6, one may consider
 22 interface objects I1 and I2 as separate interfaces of the same system and one may
 23 also consider that objects I1 and I2 plus medium M comprise the interface.
 24 Although Figs. 5 and 6 show bi-directional flow and interfaces on each side of the
 25 flow, certain implementations may only have information flow in one direction (or



1 no information flow as described below) or may only have an interface object on
 2 one side. By way of example, and not limitation, terms such as application
 3 programming or program interface (API), entry point, method, function,
 4 subroutine, remote procedure call, and component object model (COM) interface,
 5 are encompassed within the definition of programming interface.

6 Aspects of such a programming interface may include the method whereby
 7 the first code segment transmits information (where "information" is used in its
 8 broadest sense and includes data, commands, requests, etc.) to the second code
 9 segment; the method whereby the second code segment receives the information;
 10 and the structure, sequence, syntax, organization, schema, timing and content of
 11 the information. In this regard, the underlying transport medium itself may be
 12 unimportant to the operation of the interface, whether the medium be wired or
 13 wireless, or a combination of both, as long as the information is transported in the
 14 manner defined by the interface. In certain situations, information may not be
 15 passed in one or both directions in the conventional sense, as the information
 16 transfer may be either via another mechanism (e.g. information placed in a buffer,
 17 file, etc. separate from information flow between the code segments) or non-
 18 existent, as when one code segment simply accesses functionality performed by a
 19 second code segment. Any or all of these aspects may be important in a given
 20 situation, e.g., depending on whether the code segments are part of a system in a
 21 loosely coupled or tightly coupled configuration, and so this list should be
 22 considered illustrative and non-limiting.

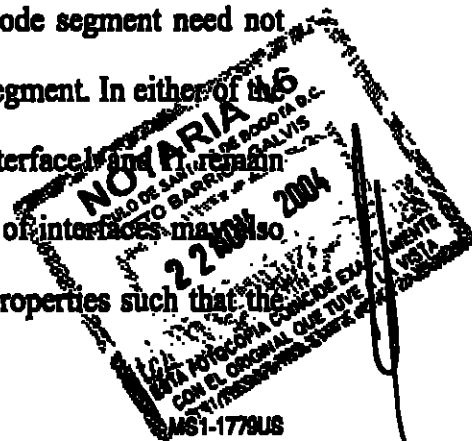
23 This notion of a programming interface is known to those skilled in the
 24 and is clear from the foregoing detailed description of the invention. Moreover,
 25 however, other ways to implement a programming interface, and, unless expressly



1 excluded, these too are intended to be encompassed by the claims set forth at the
2 end of this specification. Such other ways may appear to be more sophisticated or
3 complex than the simplistic view of Figs. 5 and 6, but they nonetheless perform a
4 similar function to accomplish the same overall result. We will now briefly
5 describe some illustrative alternative implementations of a programming interface.

7 A. FACTORING

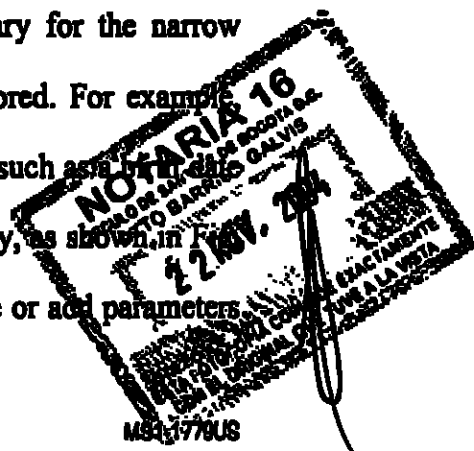
8 A communication from one code segment to another may be accomplished
9 indirectly by breaking the communication into multiple discrete communications.
10 This is depicted schematically in Figs. 7 and 8. As shown, some interfaces can be
11 described in terms of divisible sets of functionality. Thus, the interface
12 functionality of Figs. 5 and 6 may be factored to achieve the same result, just as
13 one may mathematically provide 24, or 2 times 2 times 3 times 2. Accordingly, as
14 illustrated in Fig. 7, the function provided by interface Interface1 may be
15 subdivided to convert the communications of the interface into multiple interfaces
16 Interface1A, Interface 1B, Interface 1C, etc. while achieving the same result. As
17 illustrated in Fig. 8, the function provided by interface I1 may be subdivided into
18 multiple interfaces I1a, I1b, I1c, etc. while achieving the same result. Similarly,
19 interface I2 of the second code segment which receives information from the first
20 code segment may be factored into multiple interfaces I2a, I2b, I2c, etc. When
21 factoring, the number of interfaces included with the 1st code segment need not
22 match the number of interfaces included with the 2nd code segment. In either of the
23 cases of Figs. 7 and 8, the functional spirit of interfaces Interface1 and I1 remain
24 the same as with Figs. 5 and 6, respectively. The factoring of interfaces may also
25 follow associative, commutative, and other mathematical properties such that the



factoring may be difficult to recognize. For instance, ordering of operations may be unimportant, and consequently, a function carried out by an interface may be carried out well in advance of reaching the interface, by another piece of code or interface, or performed by a separate component of the system. Moreover, one of ordinary skill in the programming arts can appreciate that there are a variety of ways of making different function calls that achieve the same result.

B. REDEFINITION

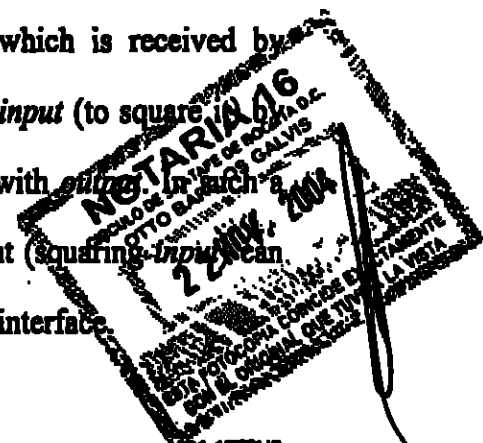
In some cases, it may be possible to ignore, add or redefine certain aspects (e.g., parameters) of a programming interface while still accomplishing the intended result. This is illustrated in Figs. 9 and 10. For example, assume interface Interfacel of Fig. 5 includes a function call *Square(input, precision, output)*, a call that includes three parameters, *input*, *precision* and *output*, and which is issued from the 1st Code Segment to the 2nd Code Segment. If the middle parameter *precision* is of no concern in a given scenario, as shown in Fig. 9, it could just as well be ignored or even replaced with a *meaningless* (in this situation) parameter. One may also add an *additional* parameter of no concern. In either event, the functionality of square can be achieved, so long as output is returned after input is squared by the second code segment. *Precision* may very well be a meaningful parameter to some downstream or other portion of the computing system; however, once it is recognized that *precision* is not necessary for the narrow purpose of calculating the square, it may be replaced or ignored. For example, instead of passing a valid *precision* value, a meaningless value such as *0* or *1* could be passed without adversely affecting the result. Similarly, as shown in Fig. 10, interface I1 is replaced by interface I1', redefined to ignore or add parameters.



1 to the interface. Interface I2 may similarly be redefined as interface I2', redefined
 2 to ignore unnecessary parameters, or parameters that may be processed elsewhere.
 3 The point here is that in some cases a programming interface may include aspects,
 4 such as parameters, that are not needed for some purpose, and so they may be
 5 ignored or redefined, or processed elsewhere for other purposes.

7 C. INLINE CODING

8 It may also be feasible to merge some or all of the functionality of two
 9 separate code modules such that the "interface" between them changes form. For
 10 example, the functionality of Figs. 5 and 6 may be converted to the functionality
 11 of Figs. 11 and 12, respectively. In Fig. 11, the previous 1st and 2nd Code Segments
 12 of Fig. 5 are merged into a module containing both of them. In this case, the code
 13 segments may still be communicating with each other but the interface may be
 14 adapted to a form which is more suitable to the single module. Thus, for example,
 15 formal Call and Return statements may no longer be necessary, but similar
 16 processing or response(s) pursuant to interface Interfacel may still be in effect.
 17 Similarly, shown in Fig. 12, part (or all) of interface I2 from Fig. 6 may be written
 18 inline into interface I1 to form interface I1". As illustrated, interface I2 is divided
 19 into I2a and I2b, and interface portion I2a has been coded in-line with interface I1
 20 to form interface I1". For a concrete example, consider that the interface I1 from
 21 Fig. 6 performs a function call square (input, output), which is received by
 22 interface I2, which after processing the value passed with input (to square
 23 the second code segment, passes back the squared result with output. In such a
 24 case, the processing performed by the second code segment (squaring input) can
 25 be performed by the first code segment without a call to the interface.



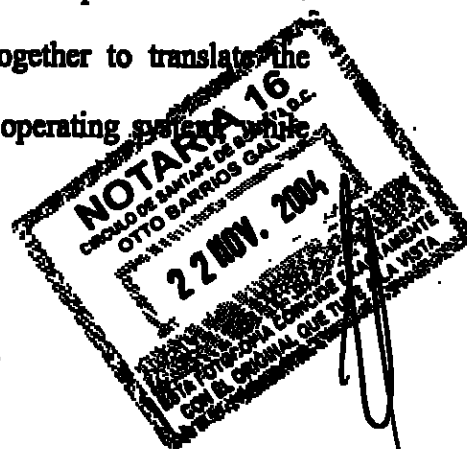
688

692

D. DIVORCE

A communication from one code segment to another may be accomplished indirectly by breaking the communication into multiple discrete communications. This is depicted schematically in Figs. 13 and 14. As shown in Fig. 13, one or more piece(s) of middleware (Divorce Interface(s), since they divorce functionality and / or interface functions from the original interface) are provided to convert the communications on the first interface, Interface1, to conform them to a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. This might be done, e.g., where there is an installed base of applications designed to communicate with, say, an operating system in accordance with an Interface1 protocol, but then the operating system is changed to use a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. The point is that the original interface used by the 2nd Code Segment is changed such that it is no longer compatible with the interface used by the 1st Code Segment, and so an intermediary is used to make the old and new interfaces compatible. Similarly, as shown in Fig. 14, a third code segment can be introduced with divorce interface DI1 to receive the communications from interface I1 and with divorce interface DI2 to transmit the interface functionality to, for example, interfaces I2a and I2b, redesigned to work with DI2, but to provide the same functional result. Similarly, DI1 and DI2 may work together to translate the functionality of interfaces I1 and I2 of Fig. 6 to a new operating system, providing the same or similar functional result.

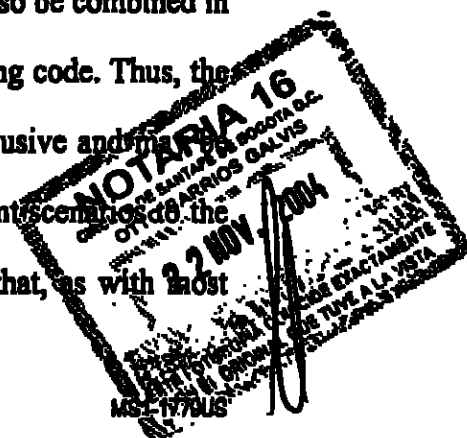
E. REWRITING



748

Yet another possible variant is to dynamically rewrite the code to replace the interface functionality with something else but which achieves the same overall result. For example, there may be a system in which a code segment presented in an intermediate language (e.g. Microsoft IL, Java ByteCode, etc.) is provided to a Just-in-Time (JIT) compiler or interpreter in an execution environment (such as that provided by the .Net framework, the Java runtime environment, or other similar runtime type environments). The JIT compiler may be written so as to dynamically convert the communications from the 1st Code Segment to the 2nd Code Segment, i.e., to conform them to a different interface as may be required by the 2nd Code Segment (either the original or a different 2nd Code Segment). This is depicted in Figs. 15 and 16. As can be seen in Fig. 15, this approach is similar to the Divorce scenario described above. It might be done, e.g., where an installed base of applications are designed to communicate with an operating system in accordance with an Interface 1 protocol, but then the operating system is changed to use a different interface. The JIT Compiler could be used to conform the communications on the fly from the installed-base applications to the new interface of the operating system. As depicted in Fig. 16, this approach of dynamically rewriting the interface(s) may be applied to dynamically factor, or otherwise alter the interface(s) as well.

It is also noted that the above-described scenarios for achieving the same or similar result as an interface via alternative embodiments may also be combined in various ways, serially and/or in parallel, or with other intervening code. Thus, the alternative embodiments presented above are not mutually exclusive and may be mixed, matched and combined to produce the same or equivalent scenarios as the generic scenarios presented in Figs. 5 and 6. It is also noted that, as with most



1 programming constructs, there are other similar ways of achieving the same or
2 similar functionality of an interface which may not be described herein, but
3 nonetheless are represented by the spirit and scope of the invention, i.e., it is noted
4 that it is at least partly the functionality represented by, and the advantageous
5 results enabled by, an interface that underlie the value of an interface.

6 7 Conclusion

8 Although the invention has been described in language specific to structural
9 features and/or methodological acts, it is to be understood that the invention
10 defined in the appended claims is not necessarily limited to the specific features or
11 acts described. Rather, the specific features and acts are disclosed as exemplary
12 forms of implementing the claimed invention.



CLAIMS

1. A programming interface embodied on one or more computer readable media, comprising:

a first group of services related to generating graphical components;

a second group of services related to binding properties of a class to a data source; and

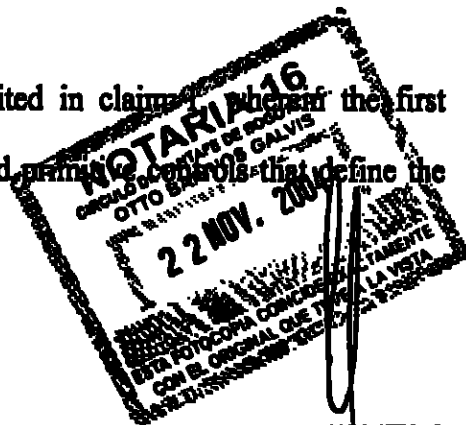
a third group of services related to formatting content.

2. A programming interface as recited in claim 1, wherein the first group of services includes a service that determines an appearance of the graphical components.

3. A programming interface as recited in claim 1, wherein the first group of services includes a service that determines a behavior of the graphical components.

4. A programming interface as recited in claim 1, wherein the first group of services includes a service that determines an arrangement of the graphical components.

5. A programming interface as recited in claim 1, wherein the first group of services includes a plurality of nested primitive controls that define the graphical components.



696

1 6. A programming interface as recited in claim 1, wherein the graphical
2 components are defined by vector graphics.

3
4 7. A programming interface as recited in claim 1, further comprising a
5 fourth group of services related to animating at least one graphical component.

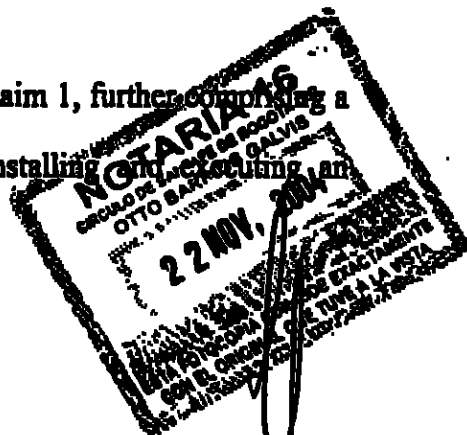
6
7 8. A programming interface as recited in claim 1, further comprising a
8 fourth group of services related to creating applications having navigation
9 capabilities.

10
11 9. A programming interface as recited in claim 1, further comprising a
12 fourth group of services related to supporting electronic ink processing systems.

13
14 10. A programming interface as recited in claim 1, further comprising a
15 fourth group of services related to combining a plurality of different media types.

16
17 11. A programming interface as recited in claim 1, further comprising a
18 fourth group of services related to executing applications on a client using a
19 browser-type interface.

20
21 12. A programming interface as recited in claim 1, further comprising a
22 fourth group of services related to automatically installing and executing an
23 application.



13. A programming interface as recited in claim 1, further comprising a fourth group of services related to serializing content.

14. A programming interface as recited in claim 1, further comprising a fourth group of services related to automating the generation of a user interface.

15. A software architecture comprising the programming interface as recited in claim 1.

16. A programming interface embodied on one or more computer readable media, comprising:

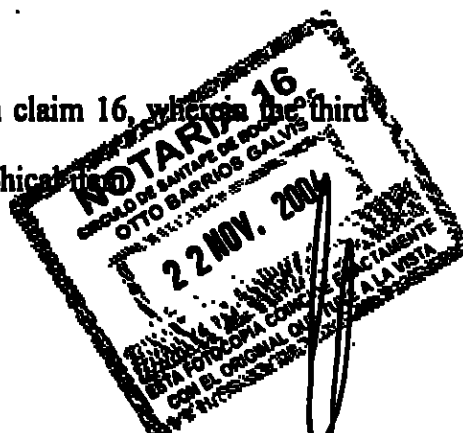
a first group of services related to formatting content prior to displaying the content;

a second group of services related to binding properties of a class to a data source; and

a third group of services related to generating imaging effects.

17. A programming interface as recited in claim 16, wherein the first group of services includes arranging a plurality of data elements.

18. A programming interface as recited in claim 16, wherein the third group of services includes animating at least one graphical element.



1 19. A programming interface as recited in claim 16, further comprising
2 a fourth group of services related to creating applications that allow a user of the
3 application to navigate between a plurality of images.

4
5 20. A programming interface as recited in claim 16, further comprising
6 a fourth group of services related to editing previously created content.

7
8 21. A programming interface as recited in claim 16, further comprising
9 a fourth group of services related to managing input received from an input device.

10
11 22. A programming interface as recited in claim 16, further comprising
12 a fourth group of services related to enabling interoperability with other
13 computing systems.



1 23. A computer system including one or more microprocessors and one
2 or more software programs, the one or more software programs utilizing an
3 application program interface to request services from an operating system, the
4 application program interface including separate commands to request services
5 comprising the following groups of services:

6 a first group of services related to generating graphical objects;

7 a second group of services related to creating components of the graphical
8 objects; and

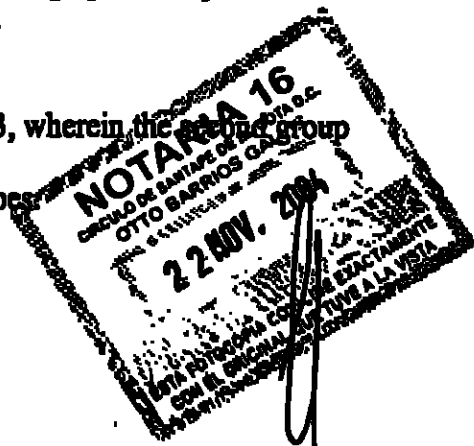
9 a third group of services related to modifying an appearance of the
10 graphical objects.

11
12 24. A computer system as recited in claim 23, wherein the first group of
13 services includes a service for defining a behavior of at least one graphical object.

14
15 25. A computer system as recited in claim 23, wherein the first group of
16 services includes a service for defining arrangement of the graphical objects.

17
18 26. A computer system as recited in claim 23, wherein modifying an
19 appearance of the graphical objects includes animating the graphical objects.

20
21 27. A computer system as recited in claim 23, wherein the second group
22 of services includes services to generate geometric shapes.



200

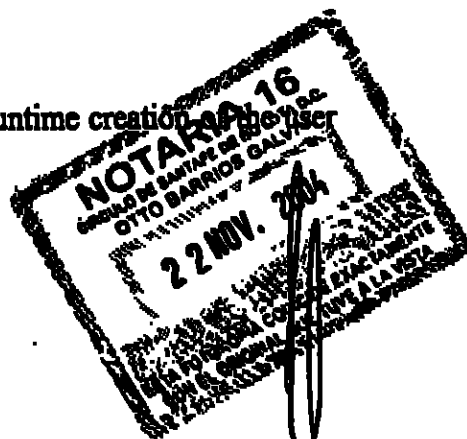
1 28. A computer system as recited in claim 23, wherein the application
2 program interface further includes a fourth group of services related to formatting
3 text.

4
5 29. A method comprising:
6 calling one or more first functions to facilitate formatting data;
7 calling one or more second functions to facilitate creating graphical objects;
8 and
9 calling one or more third functions to facilitate changing an appearance of
10 the graphical objects.

11
12 30. A method as recited in claim 29, further including calling one or
13 more fourth functions to facilitate generating a user interface using a plurality of
14 graphical objects.

15
16 31. A method as recited in claim 29, further including calling one or
17 more fourth functions to facilitate runtime creation of a user interface.

18
19 32. A method as recited in claim 29, further including:
20 calling one or more fourth functions to facilitate generating a user interface
21 using a plurality of graphical objects; and
22 calling one or more fifth functions to facilitate runtime creation of the user
23 interface.



33. A method as recited in claim 29, wherein the first functions facilitate:

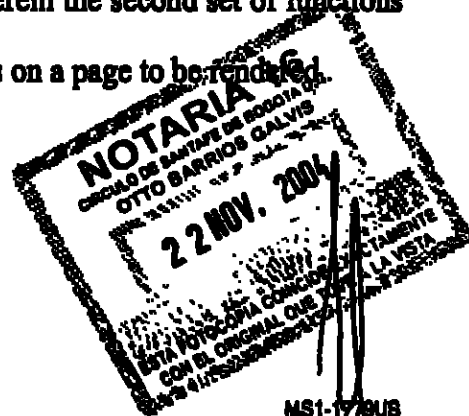
receiving user input; and
arranging data elements on a display.

34. A method as recited in claim 29, wherein the second functions facilitate generating geometric shapes.

35. A method as recited in claim 29, wherein the second functions facilitate generating at least one geometric shape and the third functions facilitate modifying an appearance of the geometric shape.

36. A system comprising:
means for exposing a first set of functions that enable creating a plurality of geometric shapes;
means for exposing a second set of functions that enable changing the manner in which the geometric shapes are arranged; and
means for exposing a third set of functions that enable modifying appearances of the geometric shapes.

37. A system as recited in claim 36, wherein the second set of functions further enable arrangement of the geometric shapes on a page to be rendered.



300

101

1 38. A system as recited in claim 36, wherein the plurality of geometric
2 shapes include a line.

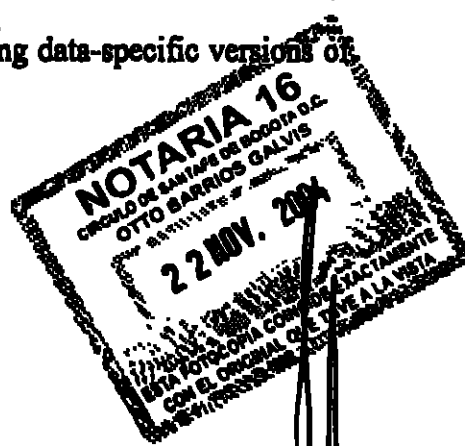
3
4 39. A system as recited in claim 36, wherein the third set of functions
5 further enable associating imaging effects with at least one geometric shape.

6
7 40. A system as recited in claim 36, wherein the third set of functions
8 further enable changing an appearance of a particular geometric shape over a
9 period of time.

10
11 41. A system as recited in claim 36, further comprising means for
12 exposing a fourth set of functions that enable generation of a user interface using
13 the plurality of geometric shapes.

14
15 42. A system as recited in claim 36, further comprising means for
16 exposing a fourth set of functions that enable associating a graphical object with
17 one or more data sources.

18
19 43. A system as recited in claim 36, further comprising means for
20 exposing a fourth set of functions that enable displaying data-specific versions of
21 graphical objects.



100

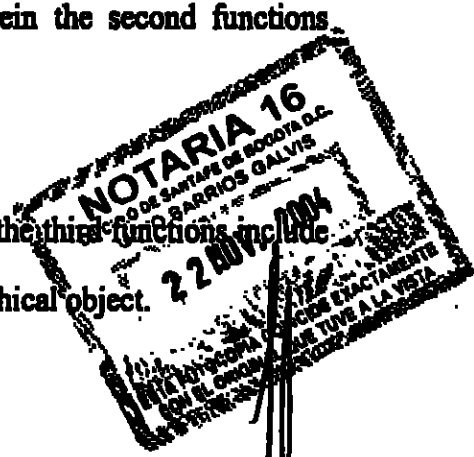
1 44. A method comprising:
2 calling one or more first functions to facilitate creating components of
3 graphical objects;
4 calling one or more second functions to facilitate generating graphical
5 objects;
6 calling one or more third functions to facilitate modifying an appearance of
7 the graphical objects;
8 calling one or more fourth functions to facilitate arranging the graphical
9 objects; and
10 calling one or more fifth functions to facilitate associating the graphical
11 objects with data sources.

12
13 45. A method as recited in claim 44, further comprising calling one or
14 more sixth functions to facilitate navigating between a plurality of displays of
15 content.

16
17 46. A method as recited in claim 44, wherein the components of the
18 graphical objects include a plurality of shapes.

19
20 47. A method as recited in claim 44, wherein the second functions
21 further facilitate generating a graphical user interface.

22
23 48. A method as recited in claim 44, wherein the third functions include
24 functions that modify the appearance of a particular graphical object.



204

1 49. A method as recited in claim 44, wherein the third functions include
2 functions that modify the appearance of one or more components of a graphical
3 object.

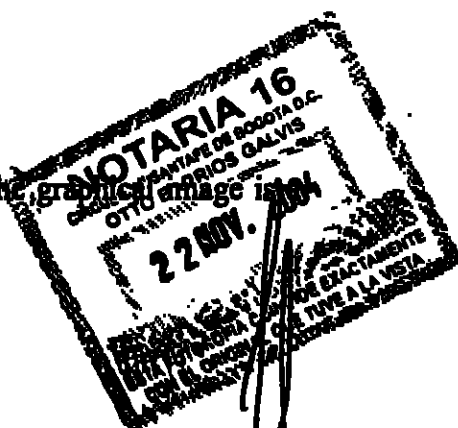
4
5 50. A method as recited in claim 44, wherein the third functions include
6 functions that move graphical objects to different positions on a display.

7
8 51. A method as recited in claim 44, wherein the third functions modify
9 an appearance of a graphical object in response to user input.

10
11 52. A method as recited in claim 44, wherein the fourth functions
12 modify an arrangement of graphical objects in response to user input.

13
14 53. A method of developing a program, the method comprising:
15 accessing a first group of functions to select components of a graphical
16 image;
17 accessing a second group of functions to generate the graphical image;
18 accessing a third group of functions to modify an appearance of at least one
19 component of the graphical image; and
20 accessing a fourth group of functions to format content associated with the
21 graphical image.

22
23 54. A method as recited in claim 53, wherein the graphical image is a
24 user interface.



205

1 55. A method as recited in claim 53, wherein the content is textual
2 information.

3
4 56. A method as recited in claim 53, wherein the content is graphical
5 data.

6
7 57. A method as recited in claim 53, further comprising accessing a fifth
8 group of functions to associate a component of the graphical image with a data
9 source.

10
11 58. A method as recited in claim 53, wherein the third group of
12 functions further modify a behavior associated with the at least one component of
13 the graphical image.



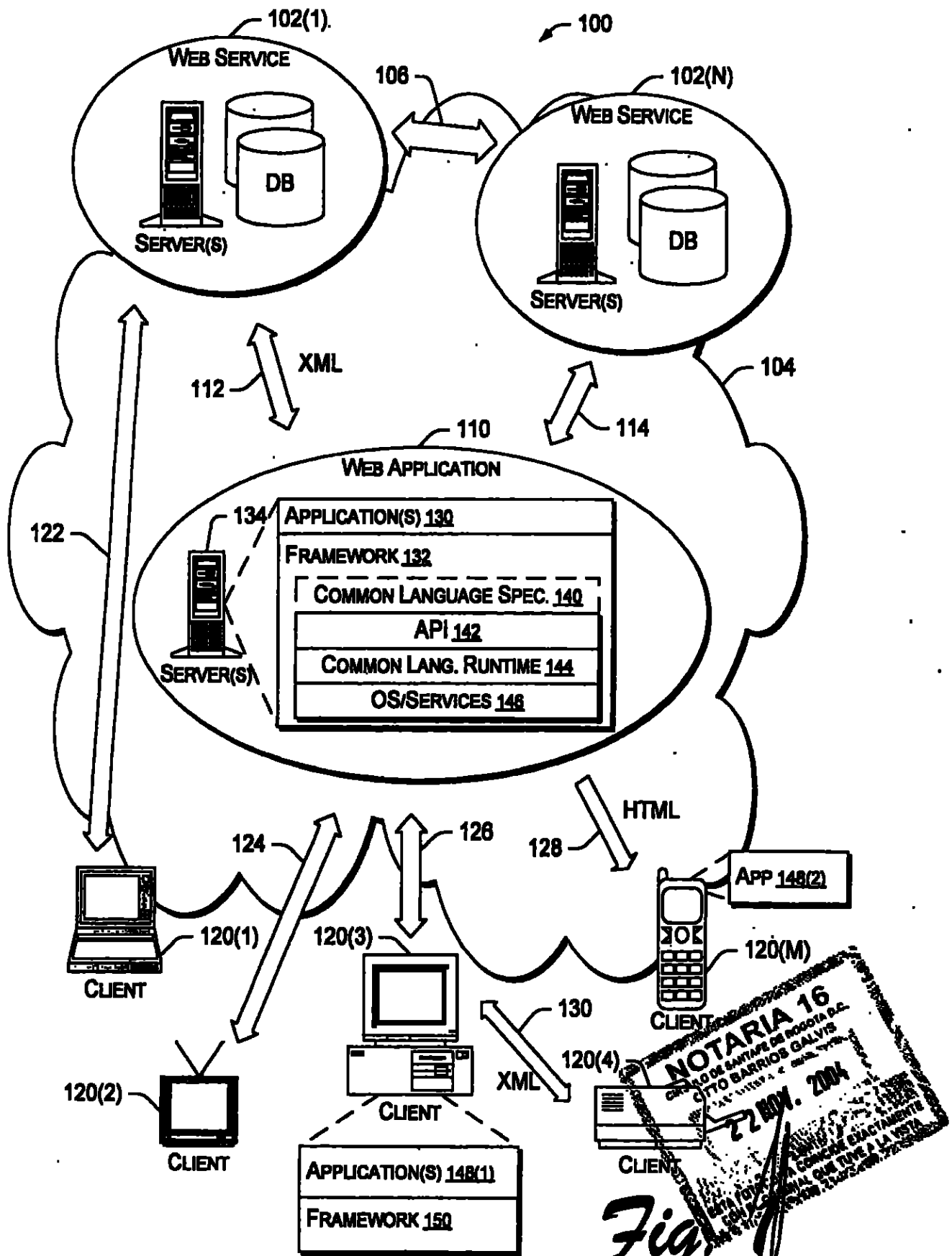
206

ABSTRACT

A programming interface provides functions for generating applications, documents, media presentations and other content. These functions allow developers to obtain services from an operating system, object model service, or other system or service.



MS1-1779US



208

MS1-1779US

132

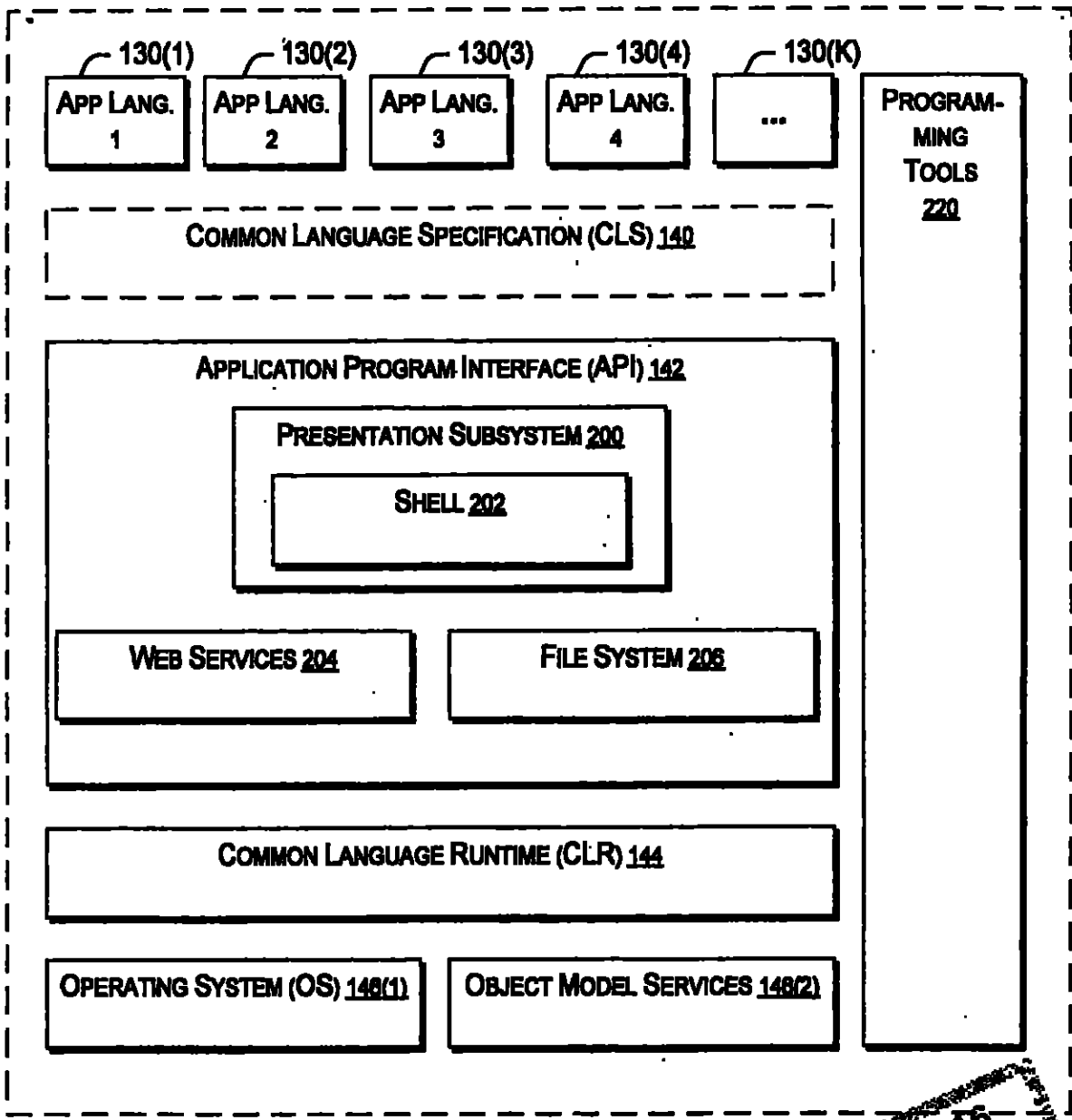
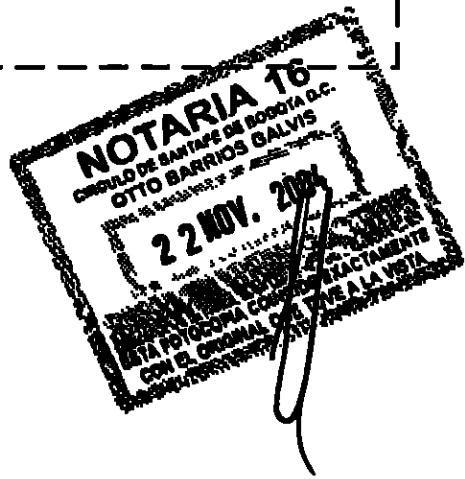
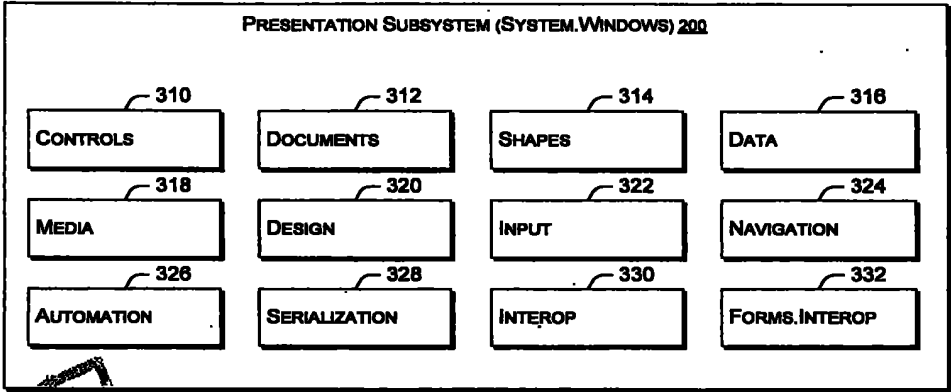


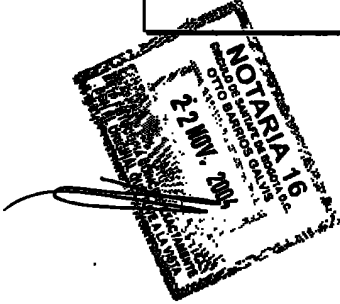
Fig. 2





MSI-179US

Fig. 3





MS1-1779US

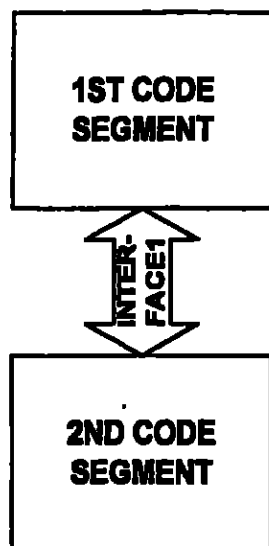


FIGURE 5

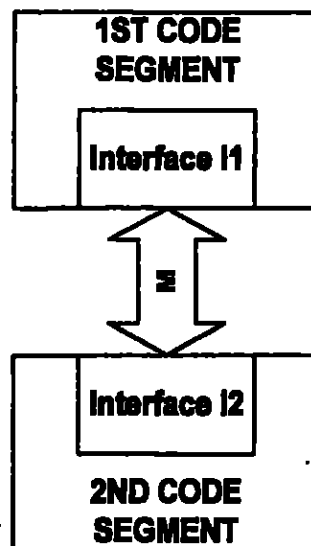


FIGURE 6

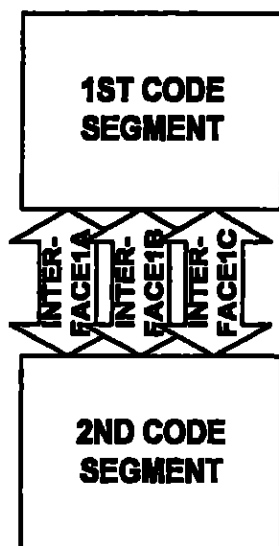


FIGURE 7

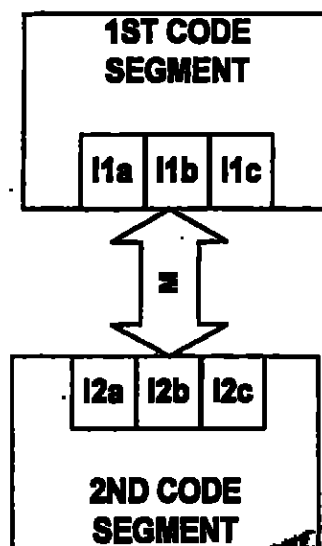
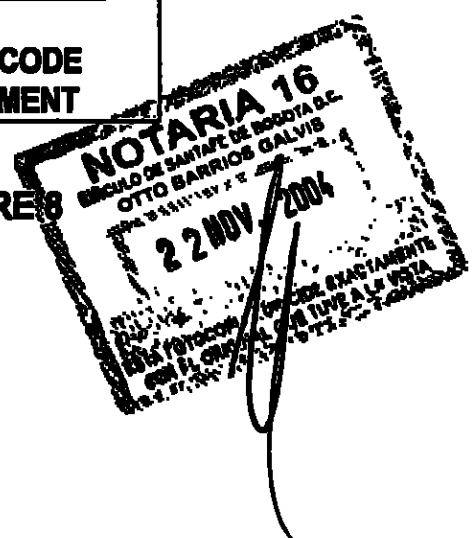


FIGURE 8



710
712

MS1-1779US

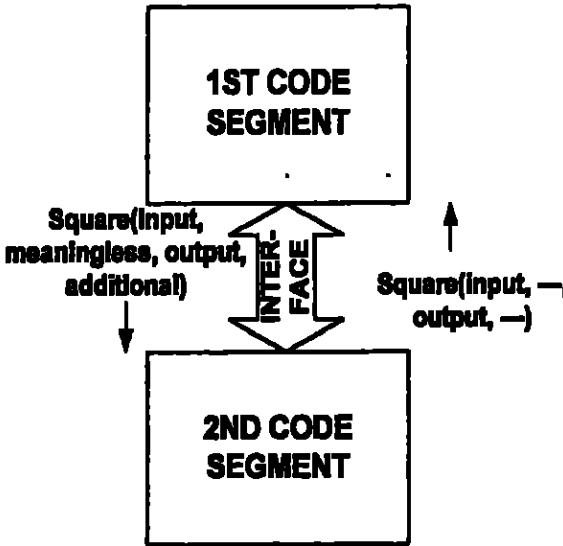


FIGURE 9

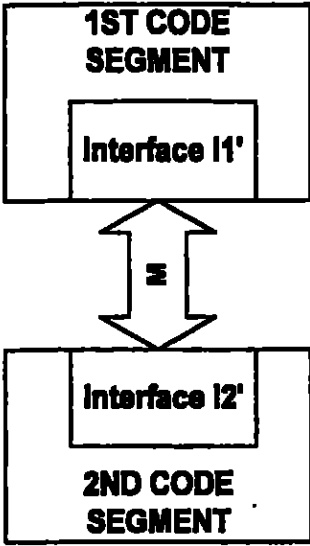


FIGURE 10

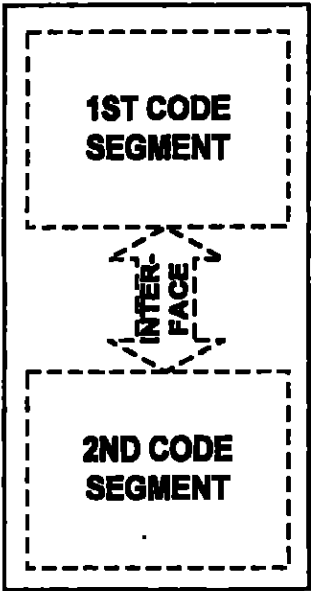


FIGURE 11

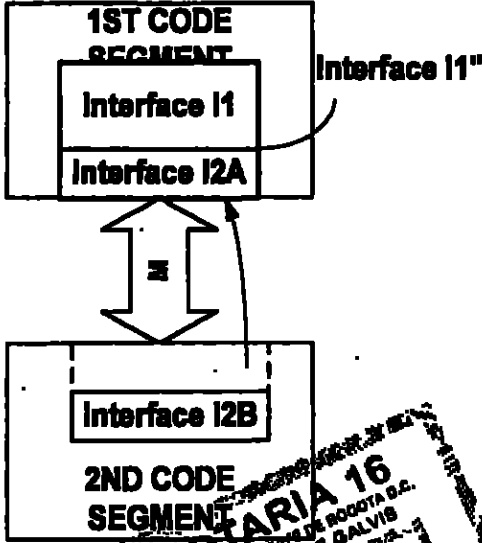


FIGURE 12

NOTARIA 16
CIRCULO DE SANTO DE RODOTA D.A.
OTTO BARRIOS GALVIS
22 NOV. 2004
ORIGINAL DE LA VISTA

110

MS1-1779US

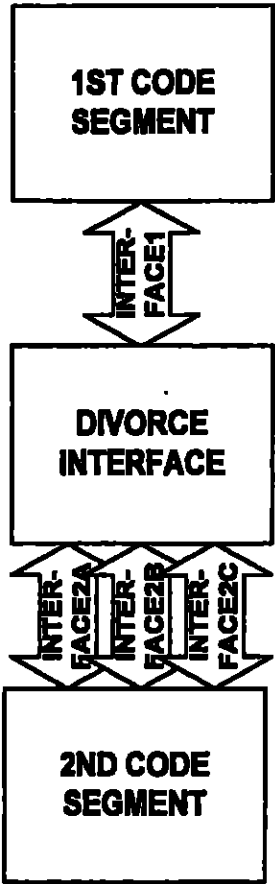


FIGURE 13

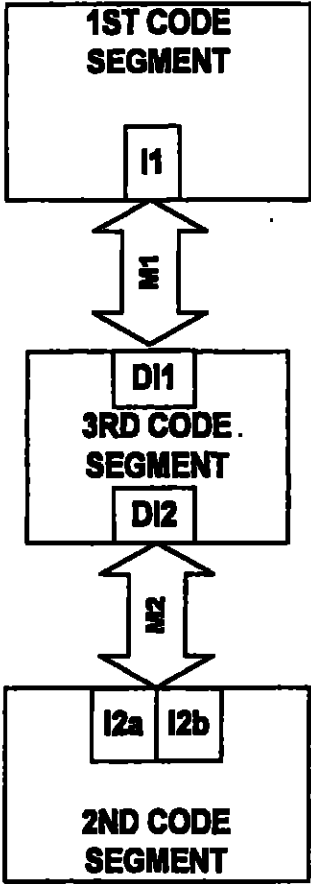


FIGURE 14



714

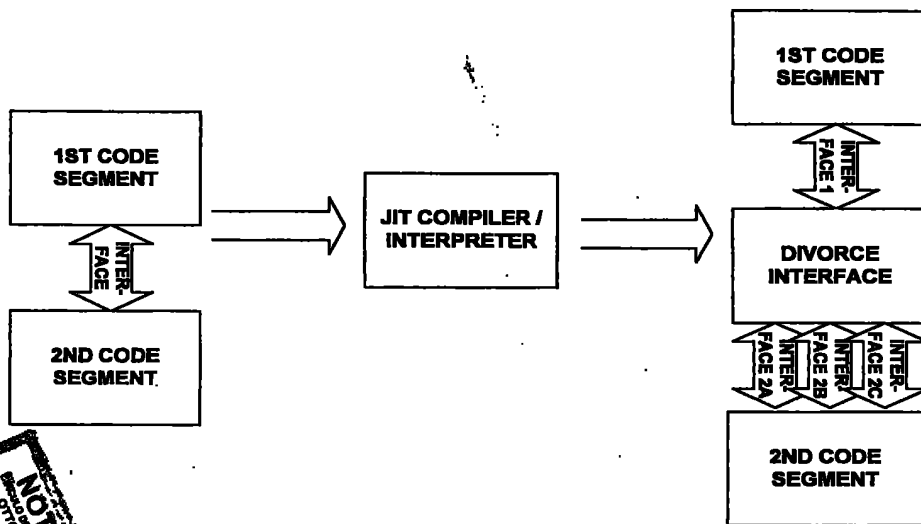
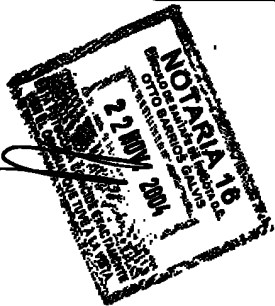


FIGURE 15

MS1-1778US



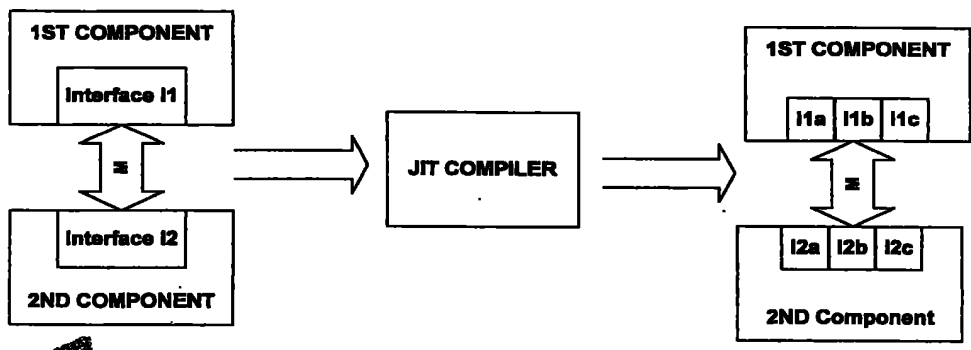
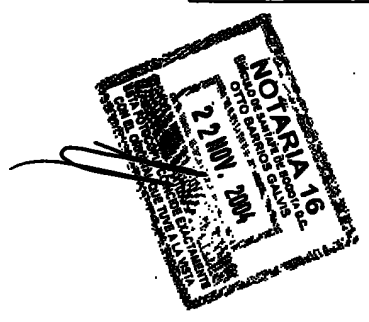


FIGURE 16

MSI-1778US



716

LOS ESTADOS UNIDOS DE AMERICA.
A todos aquellos a los que se presente este documento:

EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS

La Oficina de Patentes y Marcas de los Estados Unidos

Septiembre 16, 2004.

ESTO CERTIFICA QUE EL ANEXO A LA PRESENTE ES UNA COPIA AUTENTICA DE LOS REGISTROS DE LA OFICINA DE PATENTES Y MARCAS DE LOS ESTADOS UNIDOS DE AMERICA CORRESPONDIENTES A LOS DOCUMENTOS DE LA SOLICITUD DE PATENTE IDENTIFICADA A CONTINUACIÓN LOS CUALES CUMPLIERON CON LOS REQUERIMIENTOS PARA OTORGARLE UNA FECHA DE PRESENTACIÓN BAJO EL 35 USC 111.

NUMERO DE SOLICITUD: 10/694.080
FECHA DE SOLICITUD: Octubre 24, 2003.

(SELLO DE LA OFICINA DE PATENTES Y MARCAS DE LOS ESTADOS UNIDOS,
DEPARTAMENTO DE COMERCIO) Por la autoridad del
Comisionado de Patentes y Marcas

Firmado (Ilegible)
P. Swain
Oficial de Certificación



(Sello) Código de barras al lado del documento
04772 U.S. PTO.

(Sello) EV355227078

(Sello) Código de barras 22141USPTO
10/694080
102403

En la Oficina de Patentes y Marcas de los Estados Unidos de América

Inventor.....Relyea et al.
Solicitante.....Microsoft Corporation
No. de Archivo del abogado.....MS1-1779US
Título: Interfase de Programación para una Plataforma de Computo

CARTA DE TRANSMISION Y CERTIFICADO DE ENVIO
POR CORREO

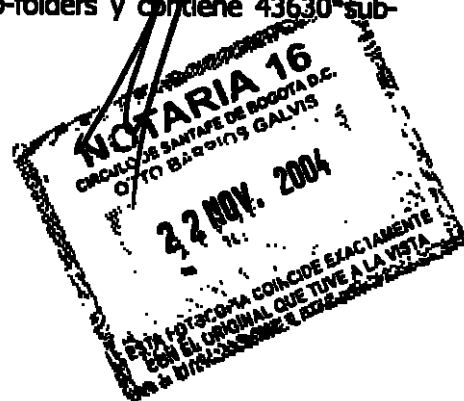
Para: Comisionado de Patentes
P.O Box 1450
Alexandria, VA 22313-1450

De: Steven R. Sponseller (Tel. 5093249256; Fax. 5093238979)
Cliente No. 22801

Los siguientes ítems enumerados acompañan esta carta de transmisión y están siendo presentados para el asunto identificado en la referencia arriba.

1. Especificación - página del título, mas 188 páginas, incluyendo 58 reivindicaciones y Abstracto.
2. Carta de Transmisión incluyendo el Certificado de Correo Expreso.
3. Nueve (9) páginas de Dibujos Formales (Figs. 1-16).
4. Carta Postal de Retorno.
5. Solicitud de suspender las reglas de acuerdo con la regla 1.183
6. Dos (2) juegos duplicados de tres (3) discos compactos (CD-Rs) cada uno etiquetado "Copia 1" y "Copia 2".

Los dos juegos de discos compactos son idénticos. Los discos compactos y los archivos allí incluidos pueden ser leídos utilizando un aparato de cómputo basado en Windows® (ej., IBM-PC, o equivalente) que ejecute un Sistema Operativo del sistema Windows® (ej., Windows®NT, Windows®98, Windows® 2000, Windows® XP, etc.). El primer disco compacto en cada juego de tres (3) discos compactos (CD1 de 3) incluye un folder de archivo denominado "lhasdk" que fue creado en octubre de 2003; es de un tamaño de 586 Mbytes, contiene 9692 subfolders, y contiene 44292 subarchivos. El segundo disco compacto en cada set de tres (3) discos compactos (CD2 de 3) incluye un folder de archivo denominado "ns" que fue creado en octubre 22 2003; es de un tamaño de 605 Mbytes, contiene 12628 sub-folders, y contiene 44934 sub-archivos. El tercer disco compacto del juego de tres (3) discos compactos (CD3 de 3) incluye un folder de archivo denominado "ns" que fue creado en octubre 22 2003; es de un tamaño de 575 Mbytes, contiene 9881 sub-folders y contiene 43630 sub-archivos.



Fecha: 10-24-2003

por firma ilegible
Steven R. Sponseller
Reg. No. 39384

Certificado de envío por correo

Por medio de la presente certifico que los ítems listados arriba como adjuntos están siendo depositados en el Servicio Postal de los Estados Unidos, bien como correo de primera clase o como Correo Expreso si el espacio para el número de Correo Expreso se completo abajo, en un sobre dirigido a el Comisionado de Patentes y Marcas, Washington D.C. 20231 en la fecha indicada abajo. Cualquier número de Correo Expreso también fue marcado en los ítems listados.

Correo Expreso No. (si es aplicable) EV355227078

Fecha: 10/24/03

por Carlyn Taylor
Carlyn Taylor

Lee & Hayes, PLLC

Copia proporcionada por USPTO de la Base de Datos de Imágenes IFW el 09/13/2004



719.

Sello No. EV355227078

**EN LA OFICINA DE PATENTES Y MARCAS DE LOS ESTADOS
UNIDOS**

SOLICITUD DE PATENTE

INTERFASE DE PROGRAMACIÓN PARA PLATAFORMA DE COMPUTO

Inventor(es):

**Rob Relyea
Jeff Bogdan**

No. de archivo del abogado MS1-1779US

Copia proporcionada por USPTO de la Base de Datos de Imágenes IFW el 09/13/2004



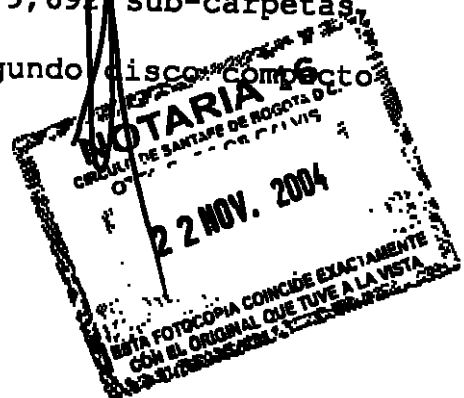
CAMPO TÉCNICO

Esta invención se relaciona con el software y el desarrollo del mismo. Más particularmente, esta invención está relacionada con una Interfaz de programación que facilita el uso de unas plataformas de software por programas de aplicación y hardware de computador.

BREVE DESCRIPCIÓN DE LOS DISCOS COMPACTOS ADJUNTOS

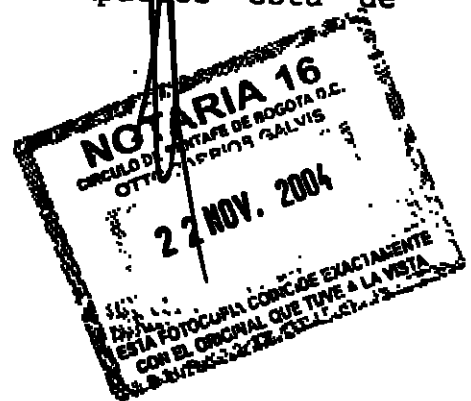
Acompañando esta especificación hay un conjunto de tres discos compactos que almacenan un Kit de desarrollo de software (SDK) para el código llamado "Longhorn" del sistema operativo Microsoft® Windows®. El SDK contiene documentación para el código llamado "Longhorn" del sistema operativo Microsoft® Windows®. Copias duplicadas de cada uno de los tres discos compactos también acompañan esta especificación.

El primer disco compacto en el conjunto de tres compact discs (CD 1 de 3) incluye una carpeta llamada "lhsdk" que fue creada en octubre 22, de 2003; esta tiene un tamaño de 586 Megabytes, contiene 9,692 sub-carpetas y contiene 44,292 sub-archivos. El segundo disco compacto



en el conjunto de tres discos compactos (CD 2 de 3) incluye una Carpeta llamada "ns" que fue creada en Octubre 22, del 2003; esta tiene un tamaño de 605 Megabytes, contiene 12,628 sub-carpetas, y contiene 44,934 sub-archivos. El tercer disco compacto en el conjunto de tres discos compactos (CD 3 de 3) incluye una carpeta llamada "ns" que fue creada e octubre 22, del 2003; esta tiene un tamaño de 575 Megabytes, contiene 9,881 sub-carpetas, y contiene 43,630 sub-archivos. Los archivos en cada uno de estos tres discos compactos pueden ser ejecutados en un dispositivo de computación basado en Windows® (por ejemplo., IBM-PC, o su equivalente) que ejecute una marca del sistema operativo Windows® (por ejemplo., Windows® NT, Windows® 98, Windows® 2000, Windows® XP, etc.). Los archivos en cada uno de los discos compactos en este conjunto de tres discos compactos son incorporados por la presente referencia.

Cada disco compacto en el conjunto de los tres discos compactos es en si mismo un CD-R, y cumple con el estándar ISO 9660. Los contenidos de cada disco compacto en el conjunto de de tres discos compactos está de



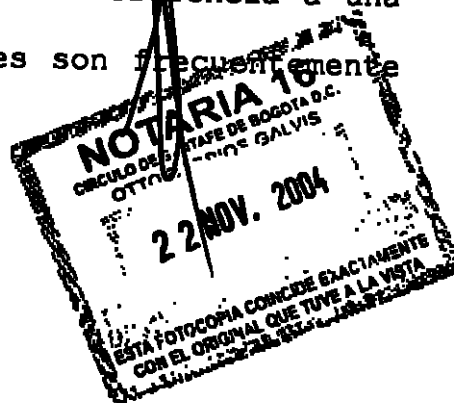
70
721

acuerdo con el Código Estándar Americano para Intercambio de Información (ASCII).

ANTECEDENTES

Desde muy temprano, el software de computador vino a ser categorizado como software de "sistema operativo" o software de "aplicación". Hablando ampliamente, una aplicación es software que permite llevar a cabo una tarea específica al usuario del computador tal como resolver una ecuación matemática o soportar un procesamiento de palabras. El sistema operativo es el software que maneja y controla el hardware del computador. La meta del sistema operativo es hacer que los recursos del computador estén disponibles para el programador de aplicaciones simultáneamente, ocultando la complejidad necesaria para realmente controlar el hardware.

El sistema operativo hace los recursos disponibles por medio de funciones que son colectivamente conocidas como la Interfaz de programa de aplicación o API. El término API también es usado para hacer referencia a una sola de esas funciones. Las funciones son frecuentemente



120

agrupadas en términos de que recurso o servicio ellas proveen para el programador de aplicaciones. El software de aplicación solicita recursos mediante el llamado de funciones individuales del API. Las funciones del API también sirven como el medio mediante el cual los mensajes y la información proveída por el sistema operativo es retransmitida al software de aplicación.

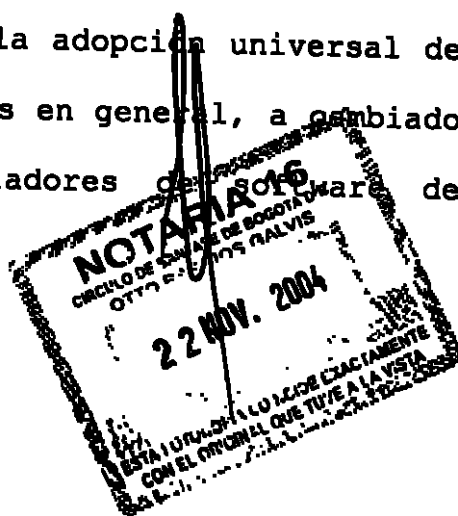
Además de cambios en el hardware, otro factor que conduce a la evolución del software del sistema operativo ha sido el deseo de simplificar y acelerar el desarrollo de software de aplicación. El desarrollo de software de aplicación puede ser una tarea atemorizante, algunas veces requiriendo de años de tiempo de un desarrollador para crear un programa sofisticado con millones de líneas de código. Para un sistema operativo popular tales como las varias versiones del sistema operativo Microsoft Windows®, los desarrolladores del software de aplicación escriben miles de diferentes aplicaciones cada año que utilizan el sistema operativo. Un sistema operativo base coherente y utilizable es requerido para soportar tantos desarrolladores de aplicaciones diversas.



A menudo, el desarrollo de software de aplicaciones puede hacerse más simple haciendo un sistema operativo más complejo. Esto es, si una función puede ser útil a diferentes programas de aplicación, puede ser mejor escribirla una sola vez para incluirla en el sistema operativo, a requerir que docenas de desarrolladores de software escriban docenas de veces la inclusión en docenas de diferentes aplicaciones. De esta manera, si el sistema operativo soporta un amplio rango de funcionalidades comunes requeridas por un número de aplicaciones, ahorros significativos pueden ser alcanzados en el costo del desarrollo del software.

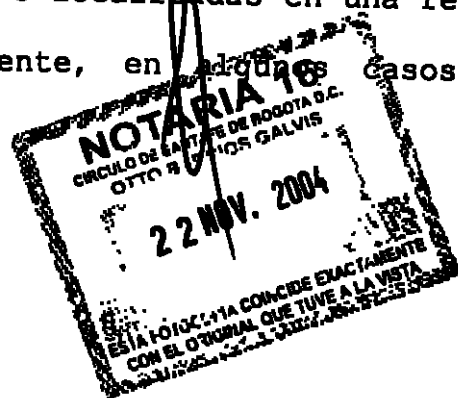
Sin considerar en donde está dibujada la línea entre el sistema operativo y el software de aplicación, es claro que para un sistema operativo útil, el API entre el sistema operativo y el hardware de computador y el software de aplicación es tan importante como la eficiencia interna de operación del sistema operativo en si mismo.

En los últimos pocos años, la adopción universal de Internet, y la tecnología de redes en general, a cambiado el panorama para los desarrolladores de software de



computador. Tradicionalmente, los desarrolladores de software se enfocaron en software de aplicaciones para un solo sitio para computadores de escritorios aislados, o computadores en una red de área local que están conectados a un número limitado de otros computadores por medio de una red de área local (LAN). Tales aplicaciones de software fueron típicamente referidas como productos "shrink wrapped" debido a que el software fue comercializado y vendido en un paquete envuelto y reducido. Las aplicaciones utilizaron APIs bien definidos para acceder al sistema operativo del computador.

En la medida en que Internet evolucionó y ganó aceptación, la industria comenzó a reconocer el poder de ubicar aplicaciones en diversos sitios de la red mundial (o simplemente la "red"). En el mundo interconectado, los clientes desde cualquier sitio podían hacer pedidos a las aplicaciones basadas en servidores ubicadas en diferentes sitios y recibir respuestas en fracciones de Segundo. Estas aplicaciones de red, sin embargo, fueron típicamente desarrolladas usando la misma plataforma del sistema operativo que fue originalmente desarrollada para máquinas de computación aisladas o localizadas en una red de computadores. Desafortunadamente, en algunos casos,



estas aplicaciones no transfieren adecuadamente a regímenes de de computación distribuida. Las plataformas subyacentes fueron simplemente no construidas con la idea de soportar ilimitadas cantidades de computadores interconectados.

Para acomodarse al cambio al ambiente de computación distribuida que aparece en Internet, la Corporación Microsoft desarrollo una plataforma de software de red conocida como ".NET" (leída como "Punto Net"). .NET de Microsoft® es software para conectar personas, información, sistema, y dispositivos. La plataforma permite a los desarrolladores crear servicios de red que podrán ejecutar sobre Internet. Este cambio de dinámica fue acompañado por un conjunto de funciones API para el marco de de trabajo de Microsoft's .NET™.

En la medida en que el marco de trabajo de .NET™ se ha venido incrementando comúnmente, maneras de incrementar la eficiencia y/o el rendimiento de la plataforma han sido identificadas.



727.

Los inventores ha desarrollado un único conjunto de funciones de API para incrementar tal eficiencia y/o rendimiento.

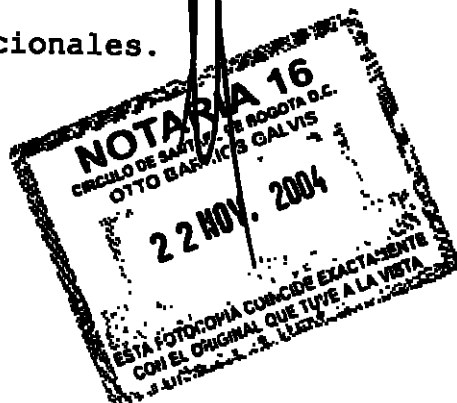
RESUMEN

Una Interfaz de programación suministra funciones para generar aplicaciones, documentos, presentaciones de medios y otro contenido. Estas funciones permiten a los desarrolladores obtener servicios de un sistema operativo, servicio de modelo de objetos, u otro sistema o servicio. En un empoderamiento, las funciones permiten a un desarrollador generar Interfaces de usuario gráficas.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Los mismos números son usados a través de los dibujos para referenciar características.

La figura 1 ilustra una arquitectura de red en la cual los clientes acceden a servicios de red sobre Internet usando protocolos convencionales.



La Figura 2 es un diagrama de bloques de una arquitectura de software para una plataforma de red, la cual incluye una Interfaz de programas de aplicación (API).

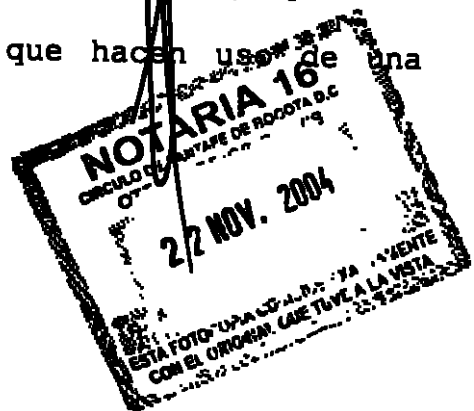
La Figura 3 es un diagrama de bloques de la presentación de subsistemas soportados por el API, así como clases de función de varias funciones de API.

La Figura 4 es un diagrama de bloques de un computador de ejemplo que puede ejecutar todo o parte de la arquitectura en software.

Las Figuras 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 y 16 ilustran varios ejemplos de implementaciones en una Interfaz de programación.

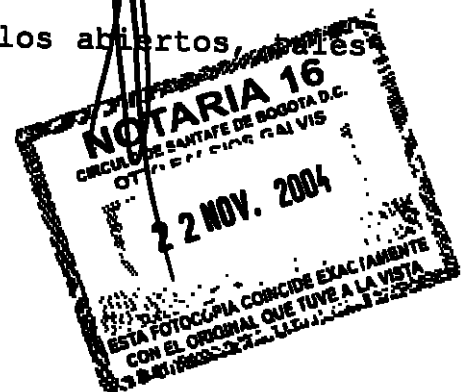
DESCRIPCIÓN DETALLADA

Esta divulgación señala una Interfaz de programa de aplicación (API) para una plataforma de red sobre la cual los desarrolladores pueden construir aplicaciones y servicios de red. Más particularmente un API ejemplar es descrito para sistemas operativos que hacen uso de una



plataforma de red, tal como el marco de trabajo de .NET™ creado por la Corporación Microsoft. El marco de trabajo de .NET™ es una plataforma de software para servicios de red y aplicaciones de red implementadas en un ambiente de computación distribuido. Representa la próxima generación de la computación de Internet, utilizando estándares de comunicación abiertos para comunicar servicios de red libremente acoplados que están colaborando para llevar a cabo una tarea particular.

En la implementación descrita, la plataforma de red utiliza XML (lenguaje de etiquetas extensibles), un estándar abierto para describir datos. XML es manejado por el Consorcio de la Red Mundial (W3C). XML es usado para describir elementos de datos en una página Web y documentos negocio al negocio. XML usa una estructura de etiqueta similar como HTML; sin embargo, mientras HTML define como son desplegados los elementos, XML define lo que contienen esos elementos. HTML usa etiquetas predefinidas, pero XML permite etiquetas definidas por el desarrollador de la página. Así, virtualmente cualquier dato puede ser identificado, permitiendo que las páginas Web funcionen como una base de datos de registros. A través del uso de XML y otros protocolos abiertos,



730

como protocolos ante objetos simples (SOAP), la plataforma de red permite la integración de un gran rango de servicios que pueden adaptarse a las necesidades del usuario. Aunque la modalidad descrita aquí es descrita en conjunción con XML y otros estándares abiertos, estos no son requeridos para la operación de la reivindicación de la invención.

Como se usa aquí, la frase Interfaz de programa de aplicación o API incluye Interfaces tradicionales que emplean llamados a métodos o funciones, así como llamados remotos (por ejemplo., relaciones entre Proxy y stub) e invocaciones SOAP/XML.

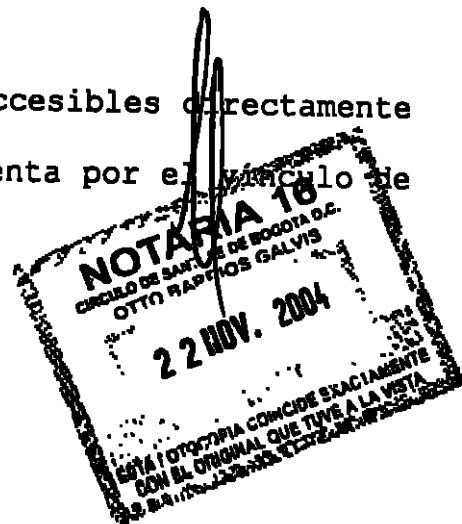
Se debe apreciar que en algunas de las descripciones de abajo, las descripciones de ciertas clases, Interfaces, enumeraciones y delegados están en blanco. Descripciones más completas a esas clases, interfaces, enumeraciones y delegados pueden encontrarse en el contenido del disco compacto que almacena el SDK referenciado arriba.

AMBIENTE DE RED DE EJEMPLO



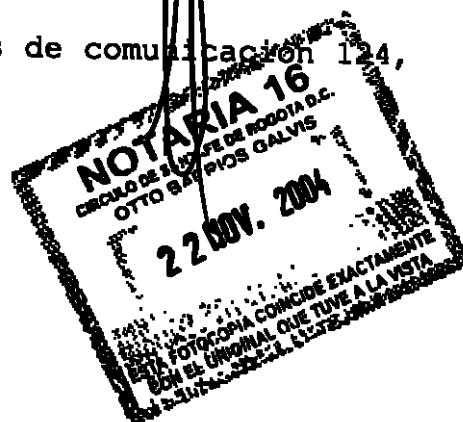
La figura 1 muestra un ambiente de red 100 en el cual una plataforma de red, tal como el marco de trabajo de .NET™, puede ser implementada. El ambiente de red 100 incluye servicios de red representativos 102(1),...,102(N), el cual suministra servicios que pueden ser accedidos sobre una red 104 (por ejemplo Internet). Los servicios de red, referenciados generalmente con el número 102, son componentes de aplicación programables que son reutilizables e interactúan programáticamente sobre la red 104, típicamente a través de la industria de protocolos de red estándar, tales como XML, SOAP, WAP (protocolos de Aplicaciones Inalámbricas) HTTP (protocolo de transporte de hipertextos), y SMTP (protocolo de transferencia de correo simple) aunque otro significado de interacción con los servicios de red sobre una red también pueden ser utilizados, tales como Llamado a Procedimientos Remotos (RPC) o tecnología de triple agente de objeto. Un servicio de red puede ser en si mismo descrito y es a menudo definido en términos de formato y ordenamiento de mensajes.

Los servicios de red 102 son accesibles directamente por otros servicios (como se representa por el vínculo de



comunicación 106) o un software de aplicación, tal como una aplicación de red 110 (como se representa por vínculos de comunicación 112 y 114). Cada servicio de red 102 es ilustrado incluyendo uno o más servidores que ejecutan software para manejar requerimientos de servicios particulares. Tales servicios frecuentemente mantienen bases de datos que almacenan información para ser servida a los que hacen las peticiones. Servicios de red pueden ser configurados para llevar a cabo alguno de una variedad de diferentes servicios. Ejemplos de servicios de red incluyen verificación de ingreso, notificación, almacenamiento en base de datos, cuotas de almacenamiento, directorios de ubicación, mapas, música, cartera electrónica, calendario/agenda, listas de teléfonos, noticias de información, juegos, tiquetes, y otros. Los servicios de red pueden ser combinados con otros y con otras aplicaciones para construir experiencias interactivas inteligentes.

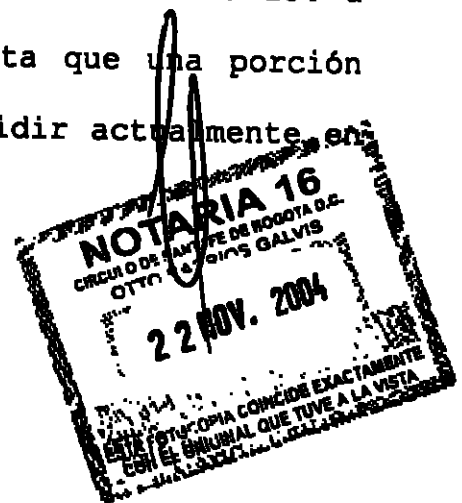
El ambiente de red 100 también incluye dispositivos clientes representativos 120(1), 120(2), 120(3), 120(4), 120(M) que utilizan el servicio de Web 102 (representado por un vínculo de comunicación 122) y/o la aplicación de Web 110 (representados como vínculos de comunicación 124,



126, y 128). Los clientes pueden comunicarse con otros usando protocolos estándar, representados por un vínculo XML ejemplar 130 entre clientes 120 (3) y 120(4).

Los dispositivos de cliente, referenciados generalmente con el número 120, pueden ser implementados de diferentes maneras. Ejemplos de posibles implementaciones de cliente incluye, sin limitación, computadores portables, computadores estacionarios, computadores personales de escritorio, cajas de televisión, dispositivos de comunicación inalámbrica, asistentes digitales personales, consolas de juegos, foto impresoras, fotocopadoras, y otros dispositivos inteligentes.

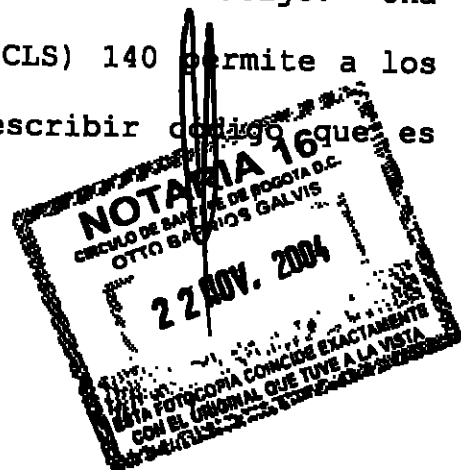
La aplicación de red 110 es una aplicación diseñada para correr en una plataforma de red y poder utilizar servicios de red 102 cuando maneja y sirve peticiones de los clientes 120. Las aplicaciones de red 110 es compuesta de uno o más software de aplicación 130 que corre arriba del marco de trabajo de programación 132, los cuales son ejecutados en uno o más servidores 134 u otros sistemas de computador. Tome nota que una porción de la aplicación de red 110 puede residir actualmente en:



uno o más clientes 120. Alternativamente, la aplicación de red 110 puede coordinarse con otro software en clientes 120 a realmente realizar estas tareas.

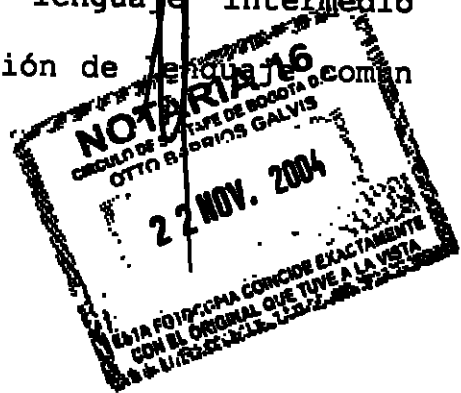
El marco de trabajo de programación 132 es la estructura que soporta la aplicación y el desarrollo de servicios por los desarrolladores de aplicaciones. Esto permite desarrollo multilenguajes e integración sutil soportando múltiples lenguajes. Soporta protocolos abiertos, tales como SOAP, y encapsula el sistema operativo y el servicio de modelo de objetos. El marco de trabajo suministra un ambiente de ejecución seguro y robusto para los múltiples lenguajes de programación y ofrece librerías de clases integradas y seguras.

El marco de trabajo 132 es una arquitectura multicapa que incluye una capa de Interfaz de programa de aplicación (API) 142, una capa de ejecución de lenguaje común (CLR) 144, y una capa de sistema y servicios operativos 146. Esta arquitectura de capas permite modificaciones y actualizaciones a varias capas sin impactar otras partes del marco de trabajo. Una especificación de lenguaje común (CLS) 140 permite a los diseñadores de varios lenguajes escribir código que es



capaz de acceder a funcionalidad de librerías. La especificación 140 funciona como un contrato entre los diseñadores de lenguaje y los diseñadores de librería que pueden usarse para promocionar interoperatividad entre lenguajes. Por adherir al CLS, librerías escritas en un lenguaje pueden ser accedidas directamente para codificar módulos escritos en otros lenguajes para alcanzar integración entre código de módulos escritos en un lenguaje y código de módulos escritos en otro lenguaje. Una implementación ejemplar detallada de un CLS es descrita en el estándar ECMA creado por los participantes en ECMA TC39/TG3. El lector es disecionado al sitio Web de ECMA en www.ecma.ch.

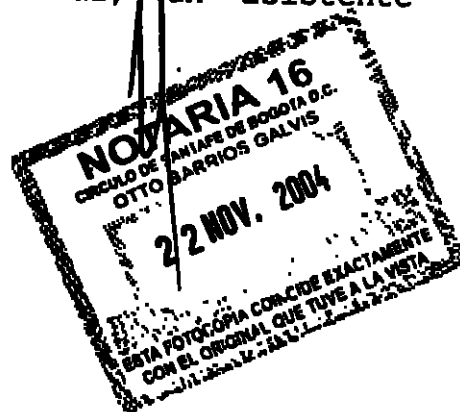
La capa de API 142 presenta grupos de funciones que las aplicaciones 130 pueden llamar para acceder a los recursos y servicios proveídos por la capa 146. Para exponer las del API para una plataforma de re, sin necesidad de entender las interconexiones complejas de como estos recursos de red realmente operan o están disponibles. Además, las aplicaciones de red pueden ser escritas en cualquier número de lenguajes de programación, y trasladadas a un lenguaje intermedio soportado por el ambiente de ejecución de lenguaje 146.



144 e incluido como parte de la especificación de lenguaje común 140. De esta manera, la capa de API 142 puede suministrar métodos para una amplia y diversa variedad de aplicaciones.

Adicionalmente, el marco de trabajo 132 puede ser configurado para soportar llamados al API realizados por aplicaciones remotas ejecutándose remotamente desde los servidores 134 que alberga el marco de trabajo. Aplicaciones representativas 148(1) y 148(2) residiendo en los clientes 120(3) y 120(M), respectivamente, pueden usar la funciones del API haciendo llamados directamente, o indirectamente a la capa del API 142 sobre la red 104.

El marco de trabajo puede también ser implementado en los clientes. El cliente 120(3) representa la situación donde un marco de trabajo 150 es implementado en el cliente. Este marco de trabajo puede ser idéntico al marco de trabajo basado en servidor 132, o modificado por propósitos del cliente. Alternativamente, el marco de trabajo basado cliente puede condensarse en el caso en que el cliente es un dispositivo de función dedicada o limitada, como es un teléfono celular, un asistente



digital personal, un computador de mano, u otro dispositivo de comunicación o computación.

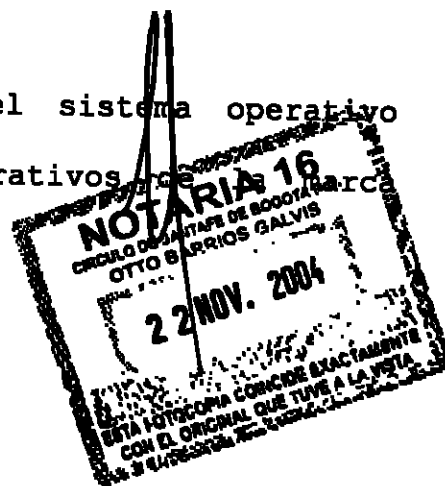
ESTRUCTURA DE PROGRAMACIÓN DE DESARROLLADORES

La Figura 2 muestra la estructura de programación 132 en más detalle. La capa de especificación de lenguaje común (CLS) 140 soporta aplicaciones escritas en una variedad de lenguajes 130 (1), 130(2), 130(3), 130(4),..., 130(K). Tales lenguajes de aplicación incluyen Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python, y otros. La especificación de lenguaje común 140 especifica un subconjunto de características o reglas sobre características que, si se presentan, permiten a varios lenguajes comunicarse. Por ejemplo, algunos lenguajes no soportan un tipo dado (por ejemplo, un tipo "int*") que por otro lado puede ser soportado por el ambiente de ejecución de lenguaje común 144. En este caso, la especificación de lenguaje común 140 no incluye el tipo. Por otro lado, los tipos que son soportados por todos o la mayoría de lenguajes (por ejemplo, el tipo "int[]") es incluido en la especificación de lenguaje común 140 para que los desarrolladores de librerías estén en libertad de usarlo y aseguren que los lenguajes puedan manejarlo.



Esto habilita la comunicación de resultados en una integración entre código de módulos escritos en un lenguaje y código de módulos escritos en otro lenguaje. Aunque diferentes lenguajes son particularmente bien ajustados a tareas particulares, la integración baja entre lenguajes permite a los desarrolladores seleccionar un lenguaje particular para un código de modulo con la habilidad de usar el código de modulo con módulos escritos en diferentes lenguajes. El ambiente de ejecución de lenguaje común 144 permite un desarrollo multilenguaje liviano, a través de herencia de lenguaje, y provee un ambiente de ejecución seguro y robusto para los múltiples lenguajes de programación. Para más información en la especificación de lenguaje común 140 y el ambiente de ejecución de lenguaje común 144, el lector es direccionado a aplicaciones co-pending tituladas "Method and System for Compiling Multiple Languages", archivada el 6/21/2000 (número serial 09/598,105) y "Unified Data Type System and Method" archivado el 7/10/2000 (número serial 09/613,289), que son incorporados por la referencia.

La estructura 132 encapsula el sistema operativo 146(1) (por ejemplo, Sistemas operativos de



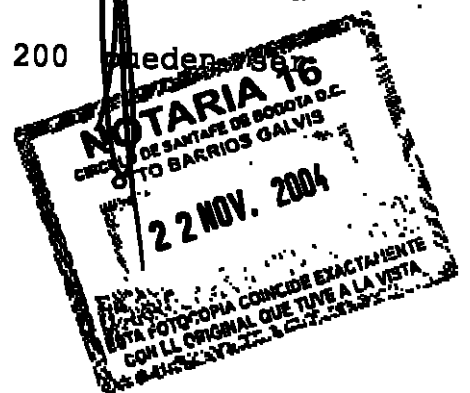
Windows®) y los servicios de modelo de objeto 146(2) (por ejemplo, el modelo de objeto componente (COM) o el COM distribuido). El sistema operativo 146 (1) provee funciones convencionales, tales como manejo de archivos, notificación, manejo de eventos, Interfaces de usuario (por ejemplo, ventanas, menú, diálogos, etc.), seguridad, autenticación, verificación, procesos e hilos, manejo de memoria y demás. Los servicios de modelo de objeto 146(2) proveen Interfaces con otros objetos para llevar a cabo varias tareas. Llamadas realizadas a la capa de API 142 son entregados a la capa de ejecución de lenguaje común 144 para ejecución local por medio del sistema operativo 146(1) y/o los servicios de modelo de objeto 146(2).

El API 142 agrupa funciones de API en múltiples espacios de nombre. Los espacios de nombre esencialmente definen una colección de clases, Interfaces, delegados, enumeraciones, y estructuras, los cuales son colectivamente llamados "tipos", que provee un conjunto específico de funcionalidad relacionada. Una clase representa una pila manejada que ubica datos que hacen referencia a semántica de asignación. Un delegado es un apuntador de función orientado a objeto. Una enumeración es una manera especial de tipo de valor que representa



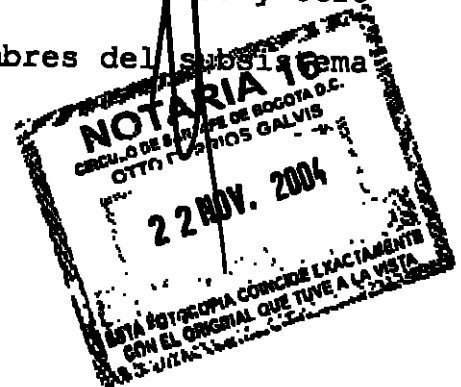
constantes nombradas. Una estructura representa datos situados estáticos que tiene un valor de semántica de asignación. Una Interfaz define un contrato que otros tipos pueden implementar.

Por usar espacio de nombres, un diseñador puede organizar un conjunto de tipos en un espacio de nombres jerárquico. El diseñador esta en capacidad de crear múltiples grupos de conjuntos de tipos, donde cada grupo contiene al menos un tipo que expone funcionalidad lógicamente relacionada. En la implementación ejemplar, el API 142 para incluir tres raíces de espacio de nombres. Se le debe anotar que aunque solo tres raíces de espacios de nombre son ilustrados en la figura 2, raíces de espacio de nombre adicionales pueden ser también incluidos en el API 142. Las tres raíces de espacio de nombre utilizadas en el API 142 son: un primer espacio de nombres 200 para un subsistema de presentación (el cual incluye un espacio de nombres 202 para un script de Interfaz de usuario), un segundo espacio de nombres 204 para servicios de red, y un tercer espacio de nombres 206 para un sistema de archivos. Cada grupo puede tener asignado un nombre. Por ejemplo, tipos en el subsistema de presentación de espacio de nombres 200 pueden ser



asignados al nombre "Windows", y tipos en el espacio de nombres del sistema de archivos 206 puede ser asignado el nombre "almacenamiento". Los nombres de grupos pueden ser organizados bajo un espacio de nombres simple "raíz global" para los APIs a nivel de sistema, tal como un sistema de espacio de nombre total. Por seleccionar y anteponer un identificador de alto nivel, los tipos en cada grupo pueden ser fácilmente referenciados por un nombre jerárquico que incluye el identificador de alto nivel antepuesto al nombre del grupo que contiene el tipo. Por ejemplo, tipos en el sistema de archive de espacio de nombre 206 pueden ser referenciados usando el nombre jerárquico "System.Storage". De esta manera, los espacios de nombres individuales 200, 204 y 206 llegan a ser las ramas más importantes del sistema de espacio de nombres y pueden transportar una designación donde los espacios de nombres individuales se les anteponen un designador tal como un prefijo "System."

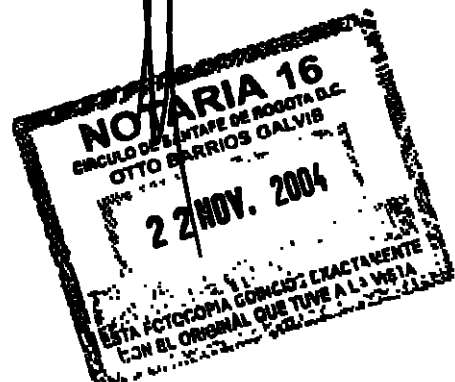
El espacio de nombre del subsistema de presentación 200 se refiere a desarrollo de contenido y programación. Este sule tipos que permiten la generación de aplicaciones, documentos, presentaciones de medios y otro contenido. Por ejemplo, espacio de nombres del subsistema



de presentación 200 suministra un modelo de programación que permite a los desarrolladores obtener servicios del sistema operativo 146(1) y/o del modelo de servicios 146(2).

El espacio de nombres de script 202 se refiere a funcionalidad de Interfaz de usuario. Este supe tipos que permiten a los desarrolladores embeber funcionalidad de Interfaz de usuario en sus aplicaciones, y por lo tanto permite a los desarrolladores extender la funcionalidad de Interfaz de usuario.

El espacio de nombres de servicios de red 204 se refiere a una infraestructura para habilitar la creación de una gran variedad de aplicaciones, por ejemplo aplicaciones simples como una aplicación de conversación que opera entre dos puntos en una intranet, y/o complejas como un servicio de red escalable para millones de usuarios. La infraestructura descrita es ventajosamente muy variable en la medida en que uno necesite solo necesite solo usar partes que son apropiadas para una solución particular compleja. La aplicación provee una fundación para construir aplicaciones basadas en mensajes de varias escalas y complejidades. La infraestructura o



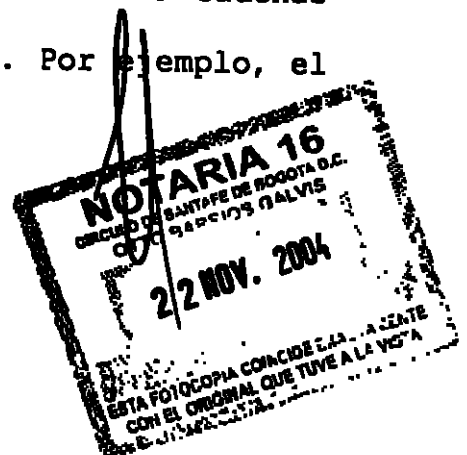
243

estructura provee APIs para mensajería básica, mensajería segura, mensajería confiable y mensajería pactada. En la modalidad descrita abajo, los APIs asociados se han factorizado en una jerarquía de espacios de nombres de manera cuidadosa para balancear utilidad, usabilidad, extensibilidad y versionabilidad. El espacio de nombres de sistema 206 se refiere a almacenamiento. Este suministra tipos que permiten que la información sea almacenada y consultada.

Adicionalmente a la estructura 132, herramientas de programación 210 son proveídas para asistir el desarrollo en la construcción de aplicaciones y/o servicios de red. Un ejemplo de herramientas de programación 210 es Visual Studio™, un juego multilenguaje de herramientas de programación ofrecido por Microsoft Corporation.

ESPACIOS DE NOMBRES DEL API RAÍZ

La Figura 3 muestra una porción del subsistema de presentación 200 en más detalle. En una modalidad, el espacio de nombres es identificado de acuerdo a la convención de nombres jerárquica en la cual las cadenas de nombres son concatenadas con periodos. Por ejemplo, el

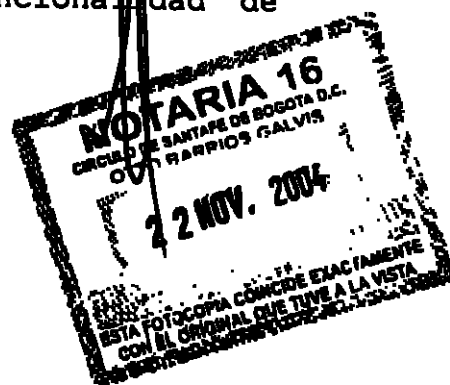


244

espacio de nombres del subsistema de presentación 200 es identificado por el nombre raíz "System.Windows". Dentro del espacio de nombres "System.Windows" hay otro espacio de nombres para varios controles identificados "System.Windows.Controls", el cual adicionalmente identifica otro espacio de nombre para primitivas (no mostrado) conocido como "System.Windows.Controls.Primitives". Con esta convención de nombramiento en mente, lo siguiente provee una descripción general de los espacios de nombre seleccionados del API 142, aunque otras convenciones de nombramientos pueden ser usadas con igual efecto.

Como se muestra en la figura 3, el subsistema de presentación 200 incluye múltiples espacios de nombres. Los espacios de nombres mostrados en la figura 3 representan una modalidad particular del subsistema de presentación 200. Otras modalidades del subsistema de presentación 200 pueden incluirse en uno o más espacios de nombres o pueden omitirse en uno o más espacios de nombres como se muestra en la figura 3.

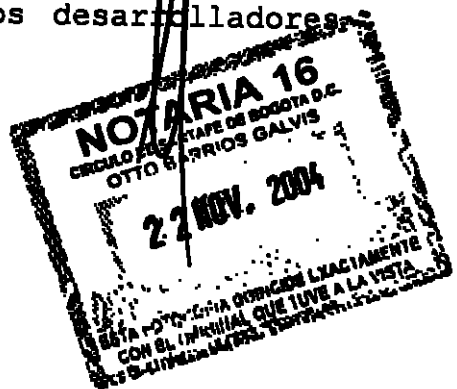
El subsistema de presentación 200 es el espacio de nombre raíz para la mayoría de las funcionalidad de



245

presentación de el API 142. El espacio de nombres de controles 310 incluye controles usados para construir información de despliegue, tal como Interfaces de usuarios y clases que permiten a un usuario interactuar con una aplicación. Controles de ejemplo incluyen "Button" que crea un botón en la pantalla, "RadioButton" que genera un botón con estilo de radio en la pantalla, "Menu" que crea un menú en la pantalla, "ToolBar" que crea una barra de herramientas en la pantalla, "Image" que genera una imagen en la pantalla y "TreeView" que crea una vista jerárquica de información.

Ciertos controles son creados para anidar y arreglar elementos múltiples. Los controles tienen un modelo lógico que ocultan los elementos usados para crear los controles, simplificando el modelo de programación. Los controles pueden ser diseñados y tematizados por un desarrollador o un usuario (por ejemplo, personalizando la apariencia y comportamiento de los botones de la Interfaz de usuario). Algunos controles tienen componentes destino que permiten un ajuste individual en el diseño de los controles individuales. Adicionalmente, los controles pueden ser subclasificados y extendidos por los desarrolladores de aplicación y los desarrolladores.



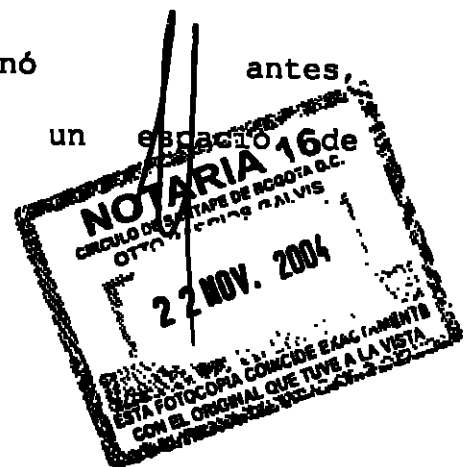
146

de componente. Los controles son representados usando vectores gráficos tal que pueda hacer cambiar el tamaño para adaptarse a los requerimientos y a una Interfaz particular u otra pantalla. Los controles están en capacidad de utilizar reanimación para realizarse, por ejemplo, la percepción interactiva de una interfaz de usuario y para mostrar acciones y reacciones.

Espacio de Nombres de Controles

El espacio de nombres de controles 310 incluye uno o más paneles, los cuales tienen controles que miden y arreglan sus hijos (por ejemplo, elementos anidados). Por ejemplo, el panel "DockPanel" cortando cada hijo por encima, la izquierda, abajo o por el lado derecho de la pantalla, y rellena el espacio sobrante con otros datos. Por ejemplo, un panel particular puede cortar menús y barras de herramientas por arriba de la pantalla, la barra de estado por abajo de la pantalla, la lista de carpetas por el lado izquierdo de la pantalla y completar el resto del espacio con una lista de mensajes.

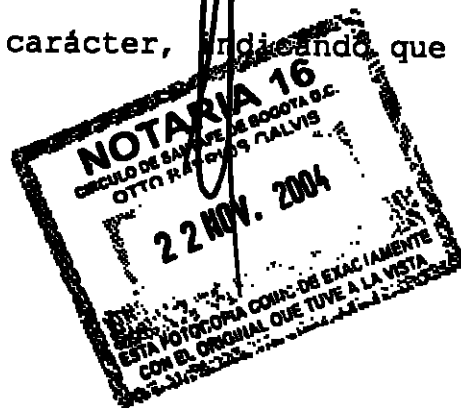
Como se mencionó antes, System.Windows.Controls.Primitives es un espacio



nombre que incluye múltiples controles que son componentes usados típicamente por los desarrolladores de controles en el espacio de nombres System.Windows.Controls.Primitives y para que los desarrolladores creen sus propios controles. Ejemplos de estos componentes incluyen "Thumb and RepeatButton". "ScrollBar", otra componente, es creada utilizando cuatro botones de repetición (uno para "line up", uno para "line down", uno para "page up", y otro para "page down") y un "Thumb" para arrastrar la vista actual en otra posición en el documento. En otro ejemplo, "ScrollViewer" es un control creado usando dos "ScrollBars" y un "ScrollArea" para suministrar el área de arrastre.

La siguiente lista contiene ejemplos de clases expuestas por el espacio de nombres System.Windows.Controls. Estas clases permiten a un usuario interactuar con, por ejemplo, una aplicación a través de varias entradas y salidas que posibilitan buenas posibilidades adicionales de despliegue.

- AccessKey - AccessKey es un elemento de la estructura que envuelve un carácter, indicando que



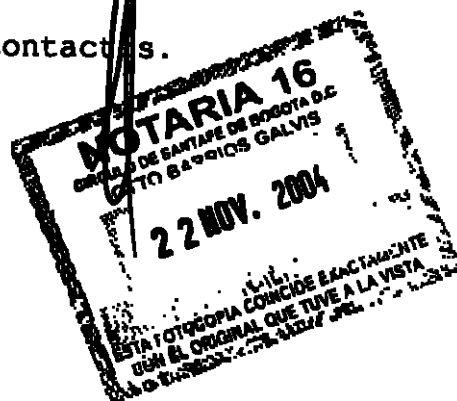
este es para recibir una indicación decorada del teclado denotando el carácter como un mnemónico de teclado. Por defecto, la indicación decorada de teclado es una línea baja.

- Audio - Elemento de Audio.
- Border - Dibuja un borde, fondo o ambos al rededor de otro elemento.
- Button - Representa una componente de botón estándar que inherentemente reacciona al evento Click.
- Canvas - Define un área dentro de la cual el usuario puede explícitamente posicionar elementos hijos por coordenadas relativas al área de Canvas.
- CheckBox - Use un CheckBox para dar al usuario una opción, tal como verdadero/falso. CheckBox permite al usuario escoger de la lista de opciones. El control CheckBox le da al usuario una combinación de opciones.
- CheckChangedEventArgs - La clase CheckChangedEventArgs contiene información adicional acerca del evento CheckChangedEvent.
- CheckStateChangedEventArgs - Esta clase CheckStateChangedEventArgs contiene información adicional acerca del evento CheckStateChangedEvent.



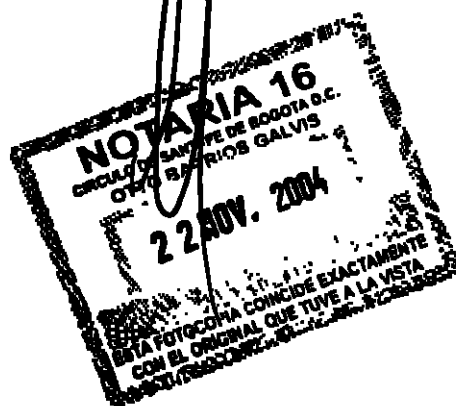
249

- ClickEventArgs - Contiene información acerca del evento Click.
- ColumnStyle - Representa un objeto cambiabile ColumnStyle.
- ColumnStyles - Objeto de patrón de lista cambiabile que es una colección de elementos cambiables.
- ComboBox - Control ComboBox.
- ComboBoxItem - Control que implementa un elemento seleccionable dentro de un ComboBox.
- ContactPickerDialog - Permite al usuario seleccionar uno o más contactos.
- ContactPropertyRequest - Permite a la aplicación solicitador información sobre una propiedad de contacto a través de un ContactPickerDialog. Esta clase no puede ser heredada.
- Colección de ContactPropertyRequest - Representa una colección de objetos ContactPropertyRequest.
- ContactSelection - Información acerca de un sistema de archivo seleccionado de Microsoft® Windows® "WinFS" o Directorio® activo de Microsoft.
- ContactSelectionCollection - Representa una colección de objetos ContactSelection.
- ContactTextBox - Un control de edición que soporta escoger contactos o propiedades de contactos.



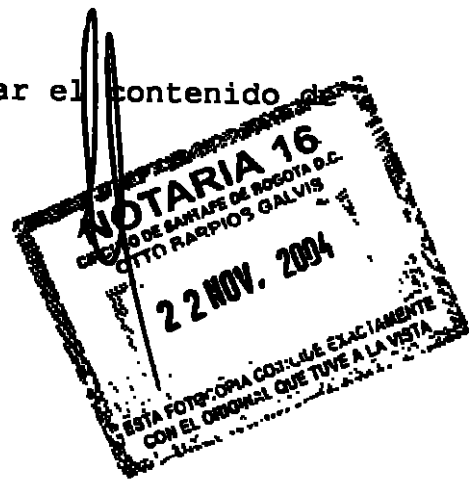
250

- `ContactTextBoxSelectionChangedEventArgs` - Argumentos para el evento `ContactTextBoxSelectionChanged`.
- `ContactTextBoxTextChangedEventArgs` - Argumentos para el evento `ContactTextBoxTextChanged`.
- `ContactTextBoxTextResolvedEventArgs` - Argumentos para el evento `TextResolvedToContact`.
- `ContentChangedEventArgs` - Argumentos de evento para `ContentChangedEvent`.
- `ContentControl` - La clase base para todos los controles con una sola pieza de contenido.
- `ContentPresenter` - `ContentPresenter` es usado dentro del diseño de un control de contenido para denotar el lugar en el árbol visual de control (chrome template) donde el contenido es adicionado.
- `ContextMenu` - Control que define un menú de selección para que el usuario lo invoque.
- `ContextMenuEventArgs` - Los datos enviados en un `ContextMenuEvent`.
- `Control` - Representa la clase base de todos los elementos interactivos de usuario. Esta clase suministra un conjunto base de propiedades para sus subclases.



251

- Decorator - Clase base para elementos que aplican efectos en y al rededor de un elemento sencillo hijo, tal como un borde.
- DockPanel - Define un área dentro de la cual usted puede hacer un arreglo de elementos hijo ya sea horizontalmente o verticalmente, relativos cada uno al otro.
- DragDeltaEventArgs - La clase DragDeltaEventArgs contiene información adicional acerca del evento DragDeltaEvent.
- FixedPanel - FixedPanel es el elemento raíz usado en documentos de formato fijo para contener páginas fijas para paginación. FixedPanel despliega contenido paginado en una página al tiempo o como un conjunto de páginas.
- FlowPanel - FlowPanel es usado para romper envolver y alinear contenido que excede la longitud de una línea sencilla. FlowPanel suministra rompimiento de línea y propiedades de alineación que puede ser usado cuando el flujo del contenido en el contenedor, por ejemplo texto, excede la longitud de una sola línea.
- Frame - Un área que puede cargar el contenido de otro árbol en aumento.



- **Generator** - Generator es el objeto que genera un UI en nombre de un `ItemsControl`, trabajando bajo la supervisión de un `GeneratorFactory`.
- **GeneratorFactory** - Un `GeneratorFactory` es el responsable de generar un UI en nombre de un `ItemsControl`. Este mantiene la asociación entre ítems en el control `ItemsCollection` (vista plana) y el correspondiente `UIElements`. El control contenedor de ítems puede preguntar la fábrica de un `Generator`, el cual hace la actual generación del UI.
- **GridPanel** - Define una área de rejilla que consiste de columnas y filas.
- **HeaderItemsControl** - La clase base para todos los controles que contienen múltiples ítems y tienen encabezado.
- **HorizontalScrollBar** - La clase `ScrollBar` horizontal.
- **HorizontalSlider** - La clase `Slider` horizontal.
- **HyperLink** - La clase `HyperLink` implementa el control de navegación. El presentador por defecto es `TextPresenter`.
- **Image** - Provee una manera fácil de incluir una imagen en un documento o en una aplicación.



- IncludeContactEventArgs - Argumentos pasados a manejadores de ContactPickerDialog. Incluye el evento IncludeContac.
- ItemCollection - Mantiene una colección de ítems discretos con un control. suministra métodos y propiedades que habilitan el cambio en el contenido de colección y obtienen datos acerca de ese contenido.
- ItemsChangedEventArgs - El evento itemsChanged es disparado por un GeneratorFactory para informar esquemas donde las conexiones de ítems han cambiado.
- ItemsControl - La clase base para todos los controles que tienen hijos múltiples.
- ItemsView - ítemsView suministra una lista plana de un ítemCollection.
- KeyboardNavigation - la clase KeyboardNavigation suministra métodos para la navegación lógica (tabulador) y direccional (flecha) entre controles enfocables.
- ListBox - Control que implementa una lista de ítems seleccionables.
- ListItem - Control que implementa un ítem seleccionable dentro de un.ListBox.



754

- **Menu** - Control que define un menú de escogencias para que el Usuario invoque.
- **MenuItem** - Un ítem hijo de un menú. Los MenuItem pueden ser seleccionados para invocar comandos. MenuItem pueden ser separadores. MenuItem pueden tener encabezados de sus menús. MenuItem pueden ser seleccionados o deseleccionados.
- **PageViewer** - Representa un control de composición de vista de documento que contiene un control de paginación, una barra de herramientas, y una barra de control de página.
- **PaginationCompleteEventArgs** - Los argumentos de evento para **PaginationCompleteEvent**.
- **PaginationProgressEventArgs** - Los argumentos de evento para el **PaginationProgressEvent**.
- **Pane** - Provee una manera de definir propiedades de ventana en un lenguaje de marcas (por ejemplo., "XAML") sin lanzar una nueva ventana.
- **Panel** - Provee una clase base para todos los elementos de panel. A fin de instanciar un elemento de panel, derivado de una clase concreta.
- **RadioButton** - **RadioButton** implementa una opción de botón con dos estados: verdadero o falso.

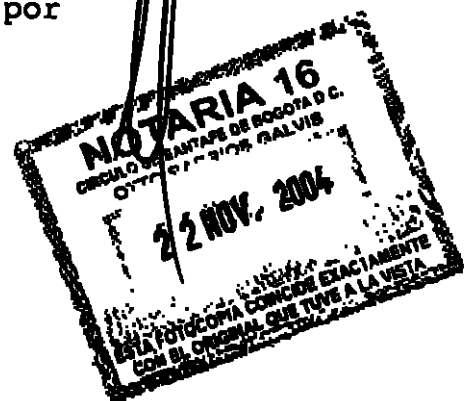


- **RadioButtonList** - Este control sirve como un control de agrupamiento de **RadioButton** y es la pieza que maneja la exclusividad entre **RadioButton** mutuos.
- El **RadioButtonList** hereda del **Selector**. El **RadioButtonList** es esencialmente un selector simple de **SelectionMode** y el concepto de selección (del **Selector**) es tecleado fuera de la propiedad **Checked** de **RadioButton** cuando es agrupado.
- **RowStyle** - Patrón cambiable y elementos cambiables.
- **RowStyles** - Objeto **IList** de patrón cambiable que es una colección de elementos cambiables.
- **ScrollChangeEventArgs** - El **ScrollChangeEventArgs** describe un cambio en el estado del avance.
- **ScrollView** - **SelectedItemCollection** - Un contenedor de un ítem seleccionado en un **Selector**.
- **SelectionChangedEventArgs** - Las entradas a un manejador de eventos de cambio de selección.
- **SimpleText** - **SimpleText** es un elemento de texto de formato simple, multi línea, liviano destinado a usarse en la intrerfas de Usuario (UI). **SimpleText** expone una variedad de propiedades del mismo formato como **Text** y puede ser usado frecuentemente para mejorar el rendimiento en el costo y alguna versatilidad.



756

- **StyleSelector** - **StyleSelector** permite al que escribe las aplicaciones suministrar un diseño propio en la lógica seleccionada. Por ejemplo, con una clase como la **Content**, usa un estilo particular para bugs **Pri1** y un estilo diferente para bugs **Pri2**. Un escritor de aplicaciones puede sobrescribir el método **SelectStyle** en una clase selector derivada y asignar una instancia de esta clase en la propiedad **StyleSelector** en una clase **ContentPresenter**.
- **Text** - Representa el control texto que habilita la representación de múltiples formatos de tex. **Text** es mayor usado dentro de una aplicación UI; escenarios de texto más avanzados se benefician de la característica adicional asignada en **TextPanel**. En muchos casos donde soporte de texto relativamente simple es requerido, **Text** es el elemento preferido debido a que es de naturaleza liviana y tiene un rango de características.
- **TextBox** - Representa el control que suministra una region editable que acepta texto de entrada.
- **TextChangedEventArgs** - La clase **TextChangedEventArgs** representas un tipo de **RoutedEventArgs** que es relevante a los eventos disparados por **TextRange.SetText()**.



257

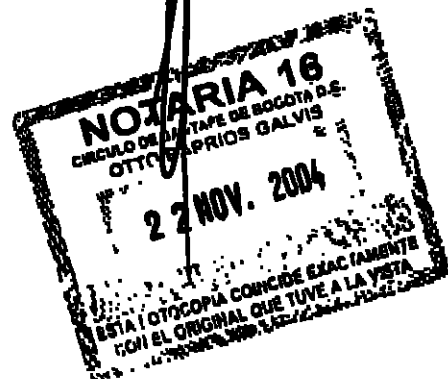
- **TextPanel** - Formatea, clasifica y dibuja textos.
TextPanel soporta múltiples líneas de texto y múltiples formatos de texto.
- **ToolTip** - Un control para desplegar información cuando el usuario vacila acerca de un control.
- **ToolTipEventArgs** - El envoi de datos en un ToolTipEvent.
- **TransformDecorator** - TransformDecorator contiene un hijo y aplica una transformada específica a él.
TransformDecorator implementa lógica para medir y arreglar los hijos en sus coordenadas locales (pre-transforma) tal como después de la transformación, el hijo se adapta estrechamente dentro del espacio del decorador y usa la máxima área. Por lo tanto el hijo no necesita tener conocimiento de que transformadas se le han aplicado a él.
- **UIElementCollection** - Un UIElementCollection es una colección ordenada de UIElements.
- **ValueChangedEventArgs** - Esta clase ValueChangedEventArgs contiene información adicional acerca del evento ValueChangedEvent.
- **VerticalScrollBar** - La clase vertical ScrollBar.
- **VerticalSlider** - La clase vertical Slider.



- Video - Ejecuta un flujo de archivo de audio o video en un rectángulo especificado dentro de las coordenadas actuales del sistema.
- VisibleChangedEventArgs - La clase VisibleChangedEventArgs contiene información adicional sobre el evento VisibleChangedEvent.

El espacio de nombres System.Windows.Controls también contiene una variedad de enumeraciones. La siguiente lista contiene un ejemplo de las enumeraciones asociadas con el espacio de nombres System.Windows.Controls.

- CharacterCase - Especifica el encajonamiento de caracteres en un control TextBox cuando el texto es digitado.
- CheckState - Especifica el estado de un control, tal como una caja de chequeo, que puede ser chequeada, deschequeada, o asignar un estado indeterminado.
- ClickMode - Especifica cuando el evento Click es disparado.



- **ContactControlPropertyPosition** - Controla la posición y despliega la propiedad del contacto.
- **ContactPickerDialogLayout** - Especifica como el **ContactPickerDialog** debe desplegar las propiedades seleccionadas.
- **ContactPropertyCategory** - Especifica cual valor tratar como el valor por defecto en caso en que la propiedad tiene múltiples valores para que el usuario escoja. Por ejemplo, si "Work" es especificado como la categoría preferida cuando se solicita la propiedad del número de telefono del **ContactPickerDialog**, y el usuario selecciona un contacto con número telefónico de oficina y de casa, el número telefónico de oficina aparece como la selección por defecto. El usuario puede entonces usar el UI para escoger el número telefónico de la casa.
- **ContactPropertyType** - Especifica una propiedad de un contacto que el **ContactPickerDialog** puede pedir al usuario.
- **ContactType** - Especifica que tipos de contactos desplegar en el **ContactPickerDialog**.



- **Direction** - Esta enumeración es usada por el **GeneratorFactory** y **Generator** para especificar la dirección en la cual el generador produce un UI.
- **Dock** - Especifica la posición del Dock de un elemento hijo dentro de un **DockPanel**.
- **GeneratorStatus** - Esta enumeración es usada por el **GeneratorFactory** para indicar su estado.
- **KeyNavigationMode** - El tipo de la propiedad **TabNavigation** especifica como el contenedor puede mover el foco cuando una navegación Tab ocurre.
- **MenuItemBehavior** - Define Los diferentes comportamientos que un **MenúItems** puede tener.
- **MenuItemType** - Define los diferentes tipos de sitios de **MenuItems**.
- **Orientation** - Tipos de orientación de deslizamiento.
- **PageViewerFit** - Selecciona como las páginas deben adaptarse dentro del area cliente **PageViewer's**.
- **PageViewerMode** - Selecciona el modo actual del **PageViewer** reflejado en el modo dropdown.
- **ScrollerVisibility** - **ScrollerVisibilty** define el comportamiento de visivilidad de una barra de desplazamiento.
- **SelectionMode** - Especifica el comportamiento de selección para el **ListBox**.

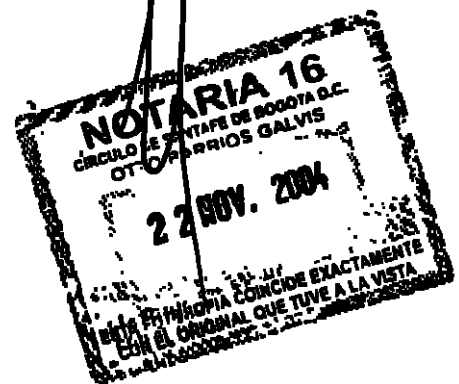


261

"Position" es una estructura de ejemplo asociada con el espacio de nombres System.Windows.Controls. Un usuario del Generator describe las posiciones usando esta estructura. Por ejemplo: Para iniciar la generación hacia adelante desde el comienzo de la lista de ítems, especifica la posición (-1, 0) y la dirección a Forward. Para iniciar la generación hacia atrás desde el final de la lista, especifica la posición (-1, 0) y la dirección Backward. Para generar los ítems después de los elementos con índice k, especifique la posición (k, 0) y la dirección Forward.

La siguiente lista contiene ejemplos de delegados asociados con el espacio de nombre System.Windows.Controls.

- CheckedChangedEventHandler - Este delegado es usado por los manejadores del evento CheckedChangedEvent.
- CheckStateChangedEventHandler - Este delegado es usado por los manejadores del evento CheckStateChangedEvent.



- ClickEventHandler - Representa los métodos que manejan el evento Click.
- ContactTextBoxSelectionChangedEventHandler - Un manejador delegado para el evento ContactTextBoxSelectionChanged.
- ContactTextBoxTextChangedEventHandler - Un manejador delegado para el evento ContactTextBoxTextChanged.
- ContactTextBoxTextResolvedEventHandler - Un manejador delegado para el evento TextResolvedToContact.
- ContentChangedDelegate - Delegado para el evento ContentChangedEvent.
- ContextMenuEventHandler - El tipo rellamado para manejar un ContextMenuEvent.
- DragDeltaEventHandler - Este delegado es usado para manejar el evento DragDeltaEvent.
- IncludeContactEventHandler - Manejador para el evento ContactPickerDialog.IncludeContact.
- ItemsChangedEventHandler - El delegado usado para manejadores que reciben el itemsChangedEventArgs.
- OpenedEventHandler - Manejador para el evento ContactPickerDialog.Opened.
- PaginationCompleteDelegate - Delegado para el PaginationCompleteEvent.



763

- `PaginationProgressDelegate` - Delegado para el `PaginationProgressEvent`.
- `ScrollChangeEventHandler` - Este delegado es usado por los manejadores del evento `ScrollChangeEvent`.
- `SelectionChangedEventHandler` - El tipo delegado para manejar un evento de cambio de selección.
- `TextChangedEventHandler` - El delegado para usar manejadores que reciben `TextChangedEventArgs`.
- `ToolTipEventHandler` - El tipo rellamado para manejar un `ToolTipEvent`.
- `ValueChangedEventHandler` - Este delegado es usado por los manejadores del evento `ValueChangedEvent`.
- `VisibleChangedEventHandler` - Este delegado es usado para manejadores del evento `VisibleChangedEventnt`.

Otro espacio de nombres, `System.Windows.Controls.Atoms`, es un sub espacio de nombres del espacio de nombres `System.Windows.Controls`. `System.Windows.Controls.Atoms` incluye controles asociados, argumentos de evento. La siguiente lista contiene un ejemplo de las clases asociadas con el espacio de nombres `System.Windows.Controls.Atoms`.



164

- PageBar - Representa un control de paginación de desplazamiento.
- PageElement - Representa una página específica de contenido de paginación. La página a ser representada es especificada por la propiedad PageSource.
- PageHoveredEventArgs - PageHoveredEventArgs suministra información acerca de donde se encuentra el apuntador del ratón.
- PageScrolledEventArgs - El PageScrolledEventArgs contiene información pertinente con el evento PageScrolled.
- PageSelectedEventArgs - El PageSelectedEvent es disparado cuando un Nuevo rango de fila/columna es realizado.
- PageSelector - PageSelector: Permite al Usuario seleccionar un rango de página de filas/columnas para ser desplegado.
- PageSource - Identifica la fuente del contenido a ser paginado. Este también suministra propiedades y métodos para formatear el contenido paginado.



La siguiente lista contiene un ejemplo de delegados asociados con el espacio de nombres System.Windows.Controls.Atoms.

- PageHoveredEventHandler - Este delegado es usado para manejar el evento PageHoveredEvent.
- PageScrolledEventHandler - Este delegado es usado para manejar el evento PageHovered.
- PageSelectedEventHandler - Este delegado es usado para manejar el evento PageSelectedEvent

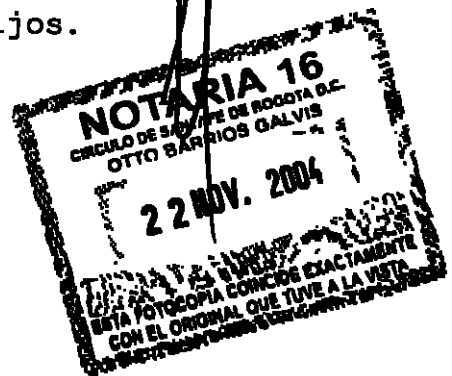
El espacio de nombres System.Windows.Controls.Primitives es otro subespacio de nombres del espacio de nombres System.Windows.Controls. Como se mencionó arriba, las primitivas del subespacio de nombres incluyen controles que se destinan para ser usados como primitivas por otros controles más complejos. La siguiente lista contiene ejemplos de clases asociadas con el espacio de nombres System.Windows.Controls.Primitives.

- ButtonBase - Cuando se sobrescribe en una clase derivada, se definen los eventos y propiedades



relevantes, y se suministra de manejadores para los eventos de entrada relevantes.

- **Popup** - Un control que crea una ventana saliente que contiene el contenido.
- **RangeBase** - Representa la clase base para elementos que tienen un rango específico. Ejemplos de estos elementos son barras de desplazamiento y barras de progreso. Esta clase define los eventos y propiedades relevantes, y provee manejadores para los eventos.
- **RepeatButton** - El control RepeatButton adiciona semánticas de repetición de cuando el evento Click ocurre.
- **ScrollArea** - ScrollArea es el elemento efectivo para el desplazamiento. El tiene contenido que fija y provee propiedades para exponer el contenido compensado y extendido. Este también suministra las entradas por defecto manejando de tal manera que el desplazamiento pueda ser manejado programáticamente o por medio del teclado o del Mouse.
- **ScrollBar** - La clase de ScrollBar.
- **Selector** - La clase base para los controles que seleccionan ítems de entre sus hijos.



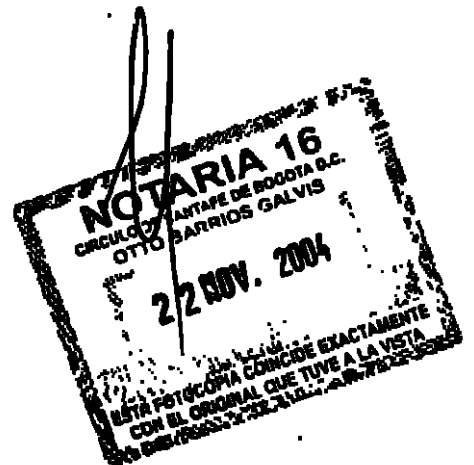
2.67

- Slider - La clase Slider.
- Thumb - El control thumb habilita la funcionalidad básica de arrastrar y mover para las barras de desplazamiento y las ventanas que ajustan su tamaño.

"IEnsureVisible" es un ejemplo de Interfaz asociada con el espacio de nombres System.Windows.Controls.Primitives. IEnsureVisible es implementado en visualización para arrastrar/mover un hijo visible dentro de una vista.

La siguiente lista contiene ejemplos de enumeraciones asociadas con el espacio de nombres System.Windows.Controls.Primitives.

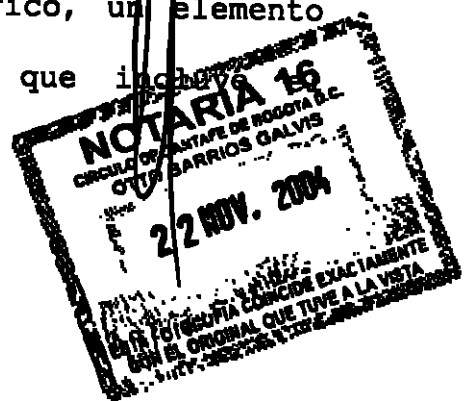
- ArrowButtonStates -
- CloseModeType - Describe como una ventana que aparece debe comportarse con varios eventos de ratón.



- **Part** - La enumeración **Part** es usada para indicar la semántica usada del control que hace la barra de desplazamiento.
- **PartStates** - **ScrollBar** parte determinada de la barra de desplazamiento.
- **PlacementType** - Describe cuando una ventana que aparece puesta en la pantalla.
- **SizeBoxStates** -

Documentos de Espacio de Nombres

Un documento de espacio de nombres 312 es una colección de semánticas y elementos de formateo que son usados para crear documentos ricos semánticamente y ricos desde el punto de vista de formateo. En una modalidad, un "elemento" es una clase que es primariamente en conjunción con una jerarquía de elementos (se refiere a un "árbol"). Estos elementos pueden ser interactivos (por ejemplo., recibir entradas de usuario por el teclado, ratón u otro dispositivo de entrada), puede presentar imágenes u objetos, y pueden asistir el arreglo de otros elementos. Elementos de ejemplo incluyen un "bloque" elemento que implementa un bloque genérico, un elemento "cuerpo" que representa el contenido que incluye



x69

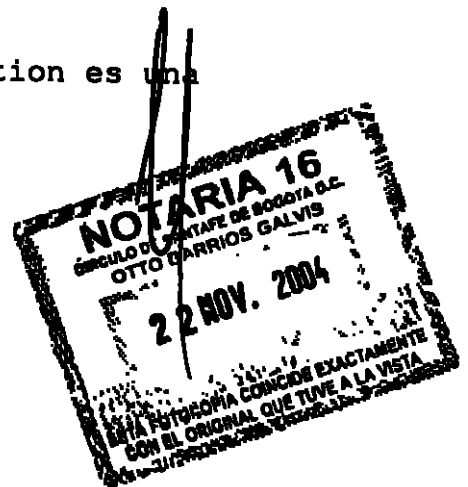
cuerpo de una tabla, un elemento "celda" que contiene datos tabulares en una tabla, un elemento "encabezado" que representa el contenido incluido en el encabezado de una tabla, y un elemento "rompimiento de página" que es usado para romper contenido a través de múltiples páginas.

La siguiente lista contiene un ejemplo de clases expuestas por el espacio de nombres System.Windows.Documents.

AdaptiveMetricsContext - AdaptiveMetricsContext provee el elemento raíz para formato de flujo adaptativo de de documentos. Una vez que un panel hijo es encapsulado en un elemento AdaptiveMetricsContext, el contenido del panel es procesado por el Reading Metrics Engine (RME). El tamaño del panel hijo es usado para calcular el número y el tamaño de cualquier columna así como también para optimizar tamaños de Fuentes y anchos de líneas.

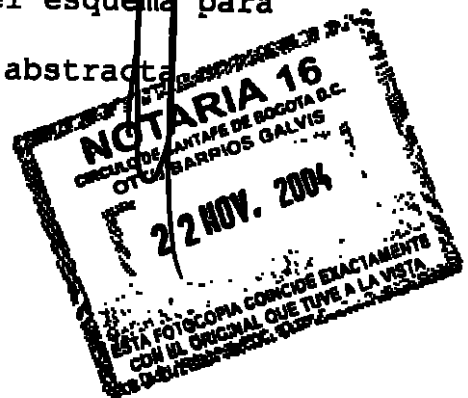


- **Block** - Implementa el elemento genérico bloque que no induce al comportamiento de representación por defecto.
- **BlockElement** - Implementa una clase base para todos los elementos **Block**.
- **Body** - Representa el contenido que comprende de un elemento cuerpo de una tabla.
- **Bold** - Implementa el elemento **Bold** derivado de **Inline**.
- **BreakRecord** - Almacena la información necesaria para continuar el contenido paginado y formateado a través de rompimientos de página. Herede de esta clase para suministrar soporte a paginación. Esta es la clase abstracta.
- **Cell** - Las **Cells** contienen datos tabulares dentro de una tabla. El elemento **Cell** es contenido dentro de un registro.
- **CellCollection** - Colección ordenada de celdas de tables.
- **Column** - El elemento **Column** es usado para distribuir el contenido de una tabla o una **Table** o un **GridPanel**.
- **ColumnCollection** - Un **ColumnCollection** es una colección ordenada de columnas.



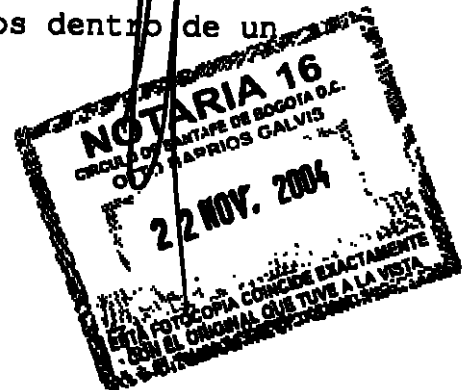
21

- **ColumnResult** - Representa la información relacionada con una columna.
- **ContainerParagraphResult** - Suministra acceso para calcular parámetros de esquema para un objeto Paragraph el cual contiene solo objetos Paragraph.
- **ContentPosition** - Representa la posición de contenido dentro de un párrafo. Herede de esta clase para describir la posición asociada al contenido. Ésta es una clase abstracta.
- **Document** - El propósito de la clase de documento es desacoplar el contenido de un documento del UI "chrome" que lo circunda. "Decoupling" significa que usted puede ser el autor de un documento sin pensar acerca de su UI (y sin comprometerse). La clase Document tiene contenido de documento, típicamente un TextPanel o un FixedPanel y sus hijos. Un árbol visual (por defecto un PageViewer) es asociado con este elemento a través del mecanismo de diseño de controles WPP.
- **DocumentPage** - Representa información de esquema para un control asociado con una página de un documento sujeto a paginación. Herede de esta clase para implementar la descripción del esquema para esos controles. Esta es una clase abstracta.



270
m

- `DocumentPageParagraphResult` - Suministra acceso a los parámetros calculados del esquema para objetos afectados por paginación.
- `FindEngine` - Clase base para algoritmos de búsqueda.
- `FindEngineFactory` - Fábrica de algoritmos Find.
- `FixedPage` - Suministra acceso a una sola página de contenido dentro del esquema de un documento formateado.
- `Footer` - Representa el contenido que comprende el pie de página de un elemento Table.
- `Header` - Representa el contenido que comprende el encabezado de un elemento Table.
- `Heading` - Implementa un elemento a nivel de bloque que representa un texto como un encabezado.
- `HyphenationDictionary` - `HyphenationDictionary` Representa un diccionario con el propósito de suministrar soporte a concatenación dentro de las aplicaciones. Este puede contener un diccionario en línea y hacer referencia a un diccionario externo. El diccionario en línea tiene una alta prioridad y puede ser aplicado antes al diccionario externo.
- `Hyphenator` - El objeto `Hyphenator` mantiene referencia a concatenación de datos dentro de un



120

HyphenationDictionary y también llevar a cabo concatenación.

- **Inline** - Implementa un elemento genérico Inline que no induce ningún comportamiento de despliegue por defecto.
- **InlineElement** - Implementa un elemento en línea genérico como clase base para todos los elementos en línea.
- **Italic** - Implementa un elemento Italic derivado de Inline.
- **LineBreak** - Representa un elemento de marca que forza un rompimiento de línea.
- **LineResult** - Suministra acceso a información calculada de una línea de texto.
- **List** - Implementa un elemento List. Listas son elementos a nivel de bloque diseñados para formatear con marcas como símbolos o numeraciones.
- **ListElementItem** - Implementa un ListElementItem, el cual soporta marcas tales como símbolos o numeración.
- **Note** - Implementa un elemento Note, el cual es análogo al elemento nota en HTML.

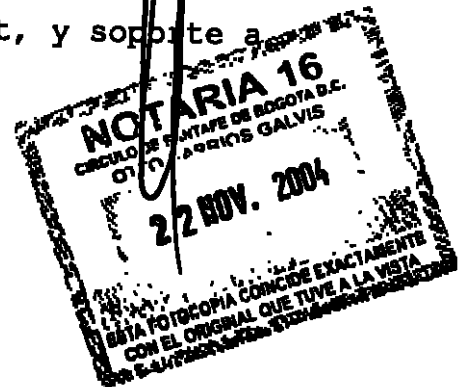


- **PageBreak** - Representa un elemento de marca usado para rompimientos de contenido a través de varias páginas.
- **PageDescriptor** - Implementa **PageDescriptor**, el cual almacena la información necesaria para crear un esquema paginado.
- **Paragraph** - Implementa un elemento a nivel de bloque usado para desplegar texto en un párrafo. El comportamiento del despliegue es análogo al de un elemento párrafo en HTML.
- **ParagraphResult** - Suministra acceso a parámetros de esquema calculados para un objeto **Paragraph**.
- **Row** - Define una fila dentro de un **GridPanel** o un elemento **Table**.
- **RowCollection** - **RowCollection** representa una colección ordenada de **Rows**.
- **RowGroup** - Especifica propiedades por defecto para un grupo de filas en un **Table** o un **GridPanel**.
- **Section** - Implementa un elemento genérico contenedor. El comportamiento de despliegue es análogo al elemento **div** en HTML.
- **SmallCaps** - Implementa un elemento **SmallCaps** en línea. **SmallCaps** son formas tipográficas que



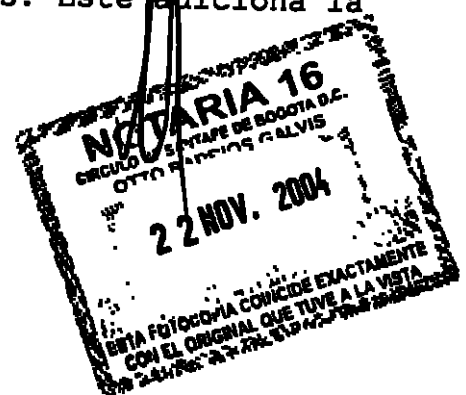
despliegan pequeñas versiones en letras mayúsculas de cartas para hacer énfasis, como en el título.

- Subscript - Representa un elemento Subscript en línea. Caracteres Subscript son escritos inmediatamente debajo, abajo a la izquierda, o abajo a la derecha de otros caracteres. Superscript - Representa un elemento Superscript en línea. Caracteres Superscript son típicamente letras o números presentadas inmediatamente abajo, abajo y a la izquierda, o abajo y a la derecha de otro caracter.
- Table - Table es usado para mostrar datos complejos en forma tabular usando un lenguaje de marcas (por ejemplo, "XAML").
- TextArray - API base para acceso texto y manipulación.
- TextChangedEventArgs - El TextChangedEventArgs define los argumentos de evento enviados mientras que el TextArray cambia.
- TextElement - TextElement provee facilidades TextRange para el TextTree. Esto es un inmutable, continuo TextRange con puntos finales fijados. Este suministra entradas ContentElement, y soporte a



Focus y Eventing. Este también provee soporte a propiedades DependencyObject.

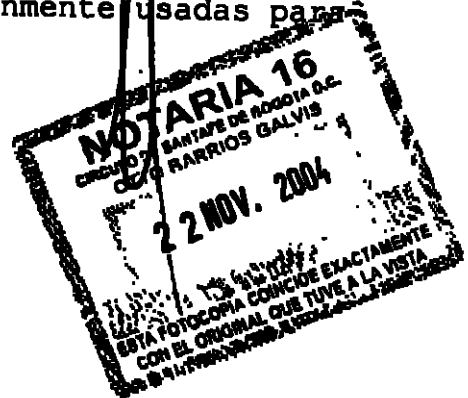
- **TextNavigator** - Este puede numerar contenido texto. Implementa una **TextPosition** movable. Este puede mover corriendo el texto o posicionándose en una posición conocida en el texto.
- **TextParagraphResult** - Suministra acceso a parámetros de esquema calculados para el texto, incluyendo objetos flotantes y figuras.
- **TextPosition** - Este es un objeto que representa cierta posición en un **TextArray**. Un objeto compacto que representa una posición en texto automáticamente mantiene la posición con los cambios en el texto. Operaciones de comparación son solamente aplicables a posiciones dentro del mismo **TextArray** (mismo contexto) **TextPosition** puede ser estático o movable. La propiedad **IsChangeable** dice el tipo de posición.
- **TextRange** - **TextRange** es una clase abstracta que provee una asociación genérica de cero o más subrangos con propiedades. La manipulación de subrangos es definida en clases derivadas.
- **TextRangeMovable** - **TextRangeMovable** es una clase abstracta para **TextRanges** movibles. Este adiciona la



272

posibilidad de mover los puntos de inicio y puntos finales basados en TextUnits.

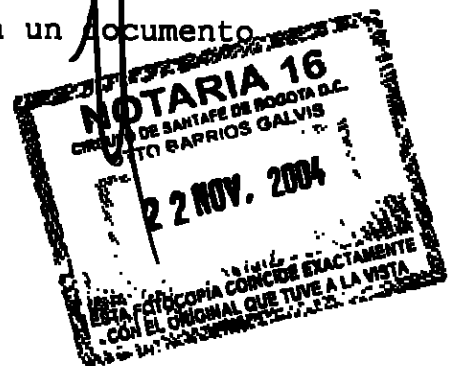
- TextTreeChangedEventArgs - El TextChangedEventArgs define argumentos de evento enviados cuando un TextArray ha cambiado.
- TextTreeDumper - TreeDumper es una clase de pruebas árbol que es pública debido a asuntos de empaquetamiento.
- TextTreeNavigator - Este es un objeto que representa ciertas posiciones movibles en un TextTree. Esta es una implementación específica de TextNavigator para usarla solamente en el TextTree.
- TextTreePosition - Este es un objeto que representa una posición incambiable en un TextTree. Esta es una implementación específica de TextPosition para usarla solamente en TextTree.
- TextTreeRange - Suministra facilidades TextRange para el TextTree. Esta es cambiante, TextRange continuos con puntos finales movibles.
- TextTreeRangeContentEnumerator - Enumerador de un objeto hijo directamente sobre un TextTreeRange.
- TextUnit - Unidad extensible de navegación de texto.
- TextUnits - Unidades de texto comúnmente usadas para TextPosition y TextRange.



- **Typography** - Suministra acceso a un conjunto enriquecido de propiedades tipográficas de OpenType.
- **UIElementParagraphResult** - EL ParagraphResult para un párrafo el cual esta compuesto completamente por UIElement. Usado por Floaters, Figuras y bloques embebidos a nivel de UIElements.
- **Underline** - Implementa un elemento Underline derivado de InlineElement.

La siguiente lista contiene Interfaces de ejemplos asociada con el espacio de nombres System.Windows.Documents.

- **IDocumentContentHost** - Implementa esta Interfaz en una ubicación de contenido tal que los hijos que son ubicados puedan notificar la ubicación cuando el contenido cambia.
- **IDocumentFormatter** - Implementa esta Interfaz para suministrar soporte a características de documentos tales como paginación.
- **ITextDocumentResult** - Implementa esta Interfaz para mantener información de columna para un documento.



279

- `ITextParagraphResult` - Implementa esta Interfaz para suministrar texto e información de posicionamiento para párrafo de texto.

La siguiente lista contiene enumeraciones de ejemplo asociadas con el espacio de nombre `System.Windows.Documents`.

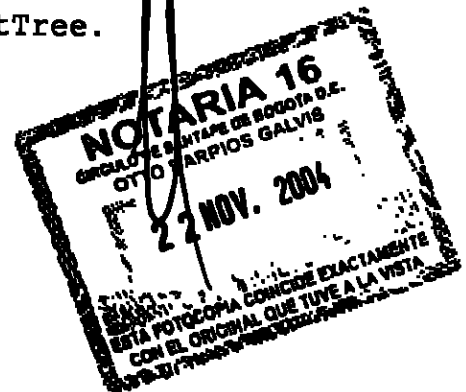
- `ElementEdge` - Esto identifica el borde de un objeto donde un `TextPosition` es localizado.
- `FindAdvancedOptions` - La opciones de búsqueda avanzadas usadas por las clases `FindAlgorithm` (inicialización de búsqueda) y `TextRangeMovable/TextSelection` (ejecución de búsqueda simplificada).
- `FindOptions` - La opción de búsqueda simplificada utilizada por los métodos de `TextBox.Find`.
- `LogicalDirection` - `LogicalDirection` define una dirección lógica para el movimiento en el texto. Este también es usado para determinar donde un `TextPosition` será movida cuanto contenido es insertado en `TextPosition`.



- `TextArrayRunType` - Esto identifica la corrida donde esta localizado una `TextPosition` teniendo en cuenta `LogicalDiretion`.
- `TextChangeOptions` - Cambios de texto posibles para `CanChangeText`.
- `TextMoveOptions` - Este controla el movimiento de `TextNavigator` especificando las condiciones para detener la navegación.

La siguiente lista contiene delegados de ejemplo asociados con el espacio de nombres `System.Windows.Documents`.

- `ObjectCloneDelegate` - Método rellamado para suministrar un clon o una copia de un `DependencyObject` cuando una parte de un `TextArray` es copiado o movido.
- `TextChangedEventHandler` - El delegado `TextChangedEventHandler` es llamado `TextChangedEventArgs` cada vez que hay contenido adicional o es removido de el `TextTree`.

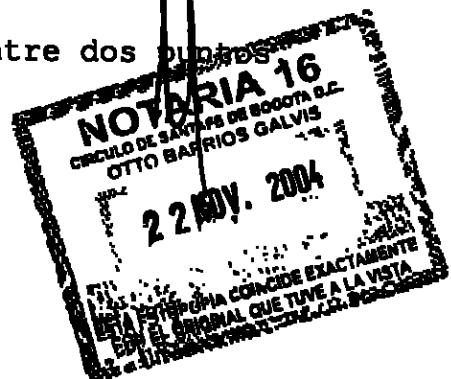


Espacios de Nombres de Forma

Un espacio de nombres de formas 714 es una colección de elementos gráficos de vector que son usados para crear imágenes y objetos. El uso de elementos gráficos de vector permite a los elementos cambiar fácilmente el tamaño para ajustarlo a los requerimientos de una Interfaz particular o un dispositivo de visualización. Elementos de ejemplo incluyen "Ellipse" elemento que dibuja una elipse, un elemento "Line" que dibuja una línea entre dos puntos, un elemento "Rectangle" que dibuja un rectángulo, y un elemento "Polygon" que dibuja un polígono como una serie de líneas conectas que forman una figura cerrada.

La siguiente lista contiene clase de ejemplo expuestas en el espacio de nombres System.Windows.

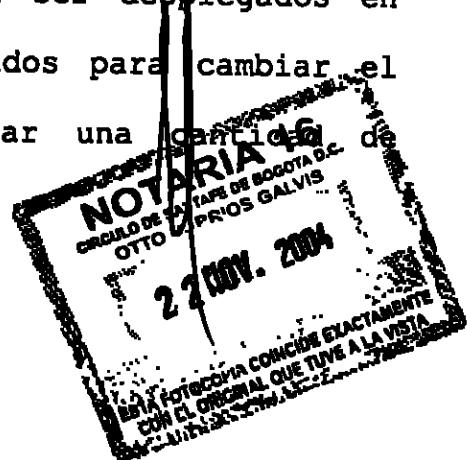
- Ellipse - Dibuja una elipse.
- Glyphs - Representa una forma de glifo en un lenguaje de marcas como "XAML". Los grifos son utilizados para representar fuentes.
- Line - Dibuja una línea recta entre dos puntos.



- Path - Dibuja una serie de líneas conectada y curvas.
- Polygon - Dibuja un polígono (una serie de líneas conectadas que forman una figura cerrada).
- Polyline - Dibuja una serie de líneas rectas conectadas.
- Rectangle - dibuja un rectángulo.
- Shape - Es una clase abstracta que suministra funcionalidad base para elementos de formas, tales como elipse, polígono y rectángulo.

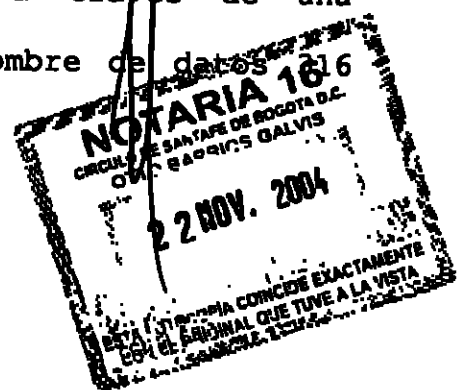
Espacio de Nombres de Datos

Un espacio de nombres de datos 316 incluye clases e Interfaces usadas para ligar propiedades de elementos a fuentes de datos, clase fuente de datos, e implementaciones específicas de datos de colecciones de datos y vistas. Esas clases e Interfaces son también utilizadas para el manejo de excepciones en entrada de datos y permiten la creación durante la ejecución de una Interfaz de usuario basada en información en varias fuentes de datos. Los datos pueden ser desplegados en forma textual o pueden ser utilizados para cambiar el formato de despliegue, como mostrar una



dólares en rojo si esta es negativa. Las clases de ejemplo incluyen una clase "Bind" que representan una declaración de objeto ligado que maneja uniones entre una propiedad dinámica de Interfaz de usuario y fuente de datos, y una clase "XmlDataSource" que sirve como fuente de datos para ligar datos a nodos de contenido XML.

Las aplicaciones orientadas por objeto típicamente representan los datos por medio de clases que definen el valor de una pieza de dato y las operaciones que pueden llevar a cabo sobre los datos. El término "data item" se refiere a uno de estos objetos. Las aplicaciones pueden manejar elementos de datos individuales o conjuntos de elementos de datos. Ellos pueden usar elementos de datos en tres maneras: (a) convirtiendo datos de fuentes externas tales como archivos de sistema, servidores remotos, bases de datos, etcétera. En un correspondiente elemento de datos en memoria, y convertir los elementos de datos modificados en la forma esperada por esas fuentes; (b) operando en los elementos de datos usando una combinación de centrado de datos y lógica de centrado de aplicación; (c) presentando el dato personificado por los elementos de datos al usuario a través de una Interfaz de usuario. El espacio de nombre de datos



suministra soporte para la primera y la tercera de estas tareas.

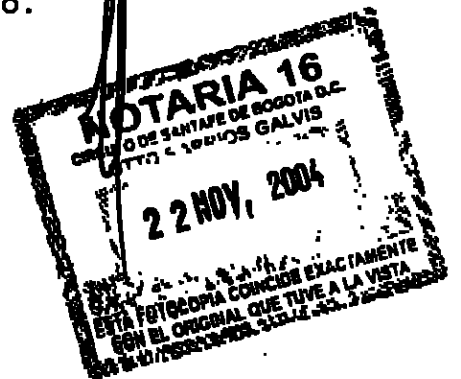
La primer tarea, obtener datos de fuentes externas, es soportado por los objetos "data source". Un objeto fuente de datos es típicamente definido como un recurso de página amplia o de aplicación amplia, y sirve como un paso a los datos. Las fuentes de datos implementan una Interfaz `IDataSource`, la cual define un mecanismo estándar por el cual las clases en el espacio de nombres de datos obtienen acceso a los datos. Un objeto particular fuente de datos implementa lógica para consultar datos actuales utilizando mecanismos apropiados para la fuente particular. En una modalidad, el espacio de nombre de dato incluye cuatro fuentes de datos:

1. `XmlDataSource`, para recobrar datos representados como XML
2. `SqlDataSource`, para recuperar datos de bases de datos SQL, tales como Microsoft SQLServer
3. `WinFSDataSource`, para recuperar datos del servicio WinFS
4. `ObjectDataSource`, para recuperar datos de un objeto arbitrario definido por la aplicación



Las aplicaciones también pueden definir sus propias clases de fuentes de datos que son suministradas a recursos de propósito especial.

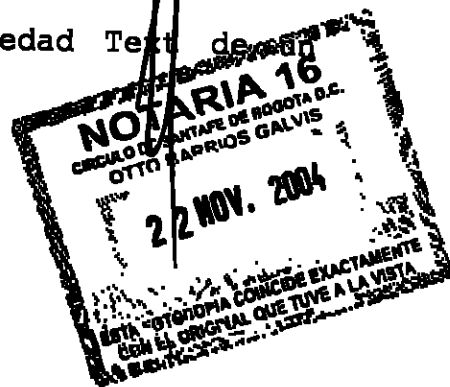
Una clase de fuente de datos es responsable de recuperar datos de una fuente externa y convertirlos en uno o más elementos de datos apropiado para ser usado por las clases ligadas. Si una colección de elementos de datos es necesitada, la aplicación puede usar cualquiera de las clases estándar de colecciones de la Estructura .Net tales como Array, ArrayList, Hashtable, etc., cualquiera de las clases de colecciones de centrado de datos del espacio de nombres System.Data tales como Dataset, o una clase de colección de centrado de datos del espacio de nombres tal como ArrayListDataCollection. Las clases mas recientes soportan notificaciones de campo; por ejemplo, cuando la aplicación cambia la colección adicionando un elemento, removiendo un elemento, ordenando la colección, etc., la colección envía una notificación. Las clases de ligado escuchan estas notificaciones y automáticamente actualizan la Interfaz del usuario para reflejar el cambio.



786

Una vez los datos han sido convertidos en elementos de datos en memoria, la aplicación puede llevar a cabo cálculos usando estos elementos y puede modificar los elementos como resultado en cálculos. Estas acciones son llevadas a cabo usando combinaciones de operaciones de centrado de datos (definidos por las clases de elementos de datos) y operaciones de centrado de aplicaciones (definidos por la aplicación de si mismo)). Las acciones pueden ser iniciadas por la aplicación automáticamente, o en respuesta a una acción del usuario. Soporte especial o cooperación entre los espacios de nombres no es requerida, por lo cual suministra una clara separación de lógica y presentación dentro de la aplicación.

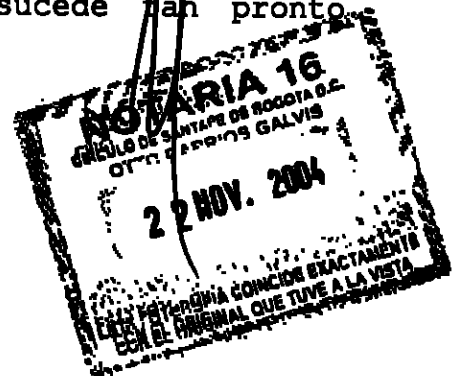
La tercera tarea relacionada con datos, presentar los datos a través de la interfaz de usuario, es soportado por las clases "binding" del espacio de nombres de datos. Estas clases habilitan la aplicación para describir la correspondencia (binding) entre una propiedad de elemento de dato (la fuente) y una propiedad de la Interfaz de usuario (el objetivo). El término data-binding (o simplemente binding) se refiere a establecer dicha correspondencia. Por ejemplo, una aplicación puede escoger el ligado de dato de la propiedad Text de un



Este tipo de correspondencia es descrita usando la clase Bind, e implementado utilizando la clase Binding. Cualquier número de propiedades de UI pueden compartir la misma descripción (Bind), pero cada propiedad tiene su propio y único Binding que mantiene el estado para una instancia particular. La descripción incluye la siguiente información a cerca de la correspondencia deseada:

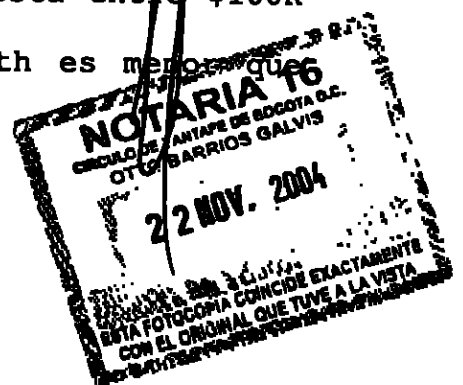
- NOTARIA 16
CIRCUITO DE SAN JOSE DE BOGOTA D.C.
OTTO GARCIA GALVIS
22 NOV. 2004
EST. POR LA QUE COINCIDE EXACTAMENTE
CON EL ORIGINAL QUE TUVE A LA VISTA

- **BindType** - cuando la correspondencia en una vía, dos vías, o uno al tiempo. En el ligado de una vía, los cambios a los elementos de datos causa actualizaciones en la propiedad de la interfaz de usuario; los datos fluyen en una vía desde la fuente hasta el objetivo. En el ligado de dos vías los datos fluyen en ambas direcciones; adicionalmente al comportamiento de una vía, cambios en la propiedad de interfaz de usuario causan actualizaciones en la propiedad del elemento de dato. En el ligado de una a la vez, la propiedad del elemento de dato es usada para inicializar la propiedad de la Interfaz de usuario, pero los cambios no son propagados en ambas direcciones.
- **Source** - una descripción de donde obtener el elemento de dato fuente. Esto puede ser del objeto fuente de dato, de otro elemento de interfaz de usuario, o del valor de la propiedad DataContext del elemento objetivo.
- **UpdateType** - cuando actualizar la propiedad fuente en un ligado de dos vías: uno en Immediate, OnLostFocus, o Explicit. En Immediate la actualización sucede tan pronto



como la propiedad de la Interfaz del usuario cambia. En OnLostFocus la actualización es retrazada hasta que el elemento destino pierda el foco del teclado- esta es apropiada para el control TextBox, para evitar el costo de actualizar después de cada tecla. En Explicit la actualización sucede cuando la aplicación explícitamente la llama.

- Transformer - un objeto que implementa la Interfaz y IDataTransformer. Este da a la aplicación una manera de modificar el valor del elemento de datos antes de usarlo en la Interfaz de usuario. La modificación puede ser una conversión simple de tipo (por ejemplo, en el ligado de una propiedad Background a la propiedad de dato BalanceOwed, la aplicación puede convertir un balance negativo a un fondo rojo y un balance positivo a verde), o una conversión específica de aplicación (por ejemplo, en un ligado de una propiedad de Text a una propiedad de dato NetWorth, la aplicación puede desplegar "Rico" si el NetWorth excede \$1M, "Burgués" si el NetWorth esta entre \$100K y \$1M, y "Pobre" si el NetWorth es menor que \$100K).



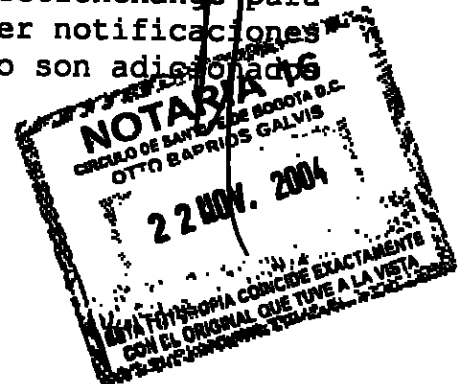
\$100K). Los transformadores son una simple pero poderosa herramienta que ayuda a separar la presentación de los datos.

Todos los ligados excepto el aun tiempo cuentan con una notificación cuando la propiedad del dato cambia, tal que el cambio correspondiente pueda ser realizado en la Interfaz de usuario. Las clases de ligado reconocen la Interfaz `IPropertyChange` (del espacio de nombres `System.ComponentModel`) como una forma de implementar las notificaciones requeridas.

Las siguientes tablas listan los miembros expuestos por el espacio de nombres `System.Windows.Data`.

Clases

<code>ArrayListCollectionView</code>	Encapsula vista colección soportada para la clase colección <code>ArrayListDataCollection</code> . Esta clase no puede ser heredada
<code>ArrayListDataCollection</code>	Suministra una implementación construida de una colección de datos de lista de arreglo con una Interfaz vista de colección fundamental. Este también implementa <code>ICollectionChange</code> para proveer notificaciones cuando son adicionados



241

elementos, eliminados elementos, o cuando toda la colección es refrescada.

Bind

Representa un objeto de declaración de ligado, usado para manejar correspondencias entre una propiedad dinámica de Interfaz de usuario (UI) y una fuente de datos

Binding

Suministra acceso a una instancia simple en ejecución de ligado. Esta clase no puede ser heredada.

BindingListCollectionView

Una clase de colección de vista usada para vistas de datos de objetos Microsoft® ActiveX® Data (ADO)

CollectionContainer

Objetos de esta clase mantienen una estructura de colección existente- por ejemplo, un ArrayListDataCollection o algún otro DataSet dentro de un itemCollection.

ContextAffinityCollectionView

Implementa una vista de colección que incluye chequeos para afinidad de contexto

DataContextObjectRef

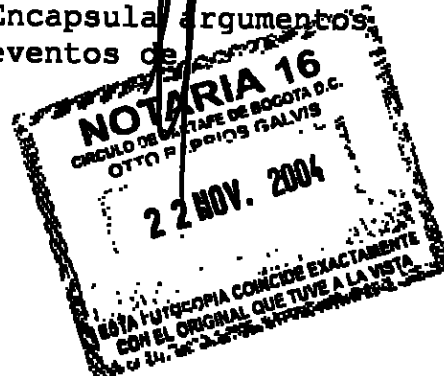
Soporta referencias de objetos a objetos que han sido usados como contextos de datos para un ligado. Esta clase no puede ser heredada.

DataSourceObjectRef

Soporta referencias de objetos fuentes de datos. Esta clase no puede ser heredada.

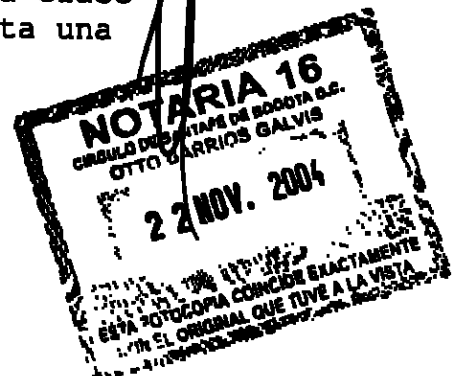
DataTransferEventArgs

Encapsula argumentos para eventos de transferencia



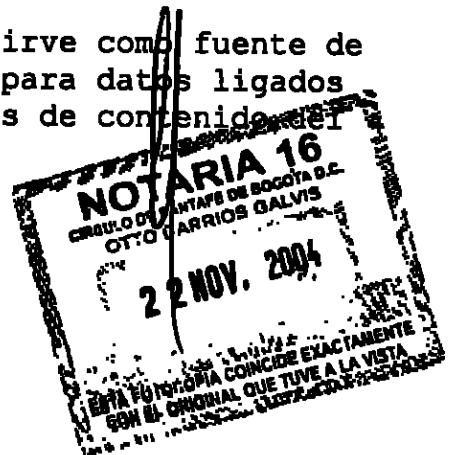
200
792

	transferencia de datos. Los eventos son eventos enrutados que son manejados específicamente por un manejador designado basado en el delegado <code>DataTransferEventHandler</code> .
<code>ElementObjectRef</code>	Representa un objeto que referencia un elemento, con el objeto en si especificado por su ID de elemento. Esta clase no puede ser heredada.
<code>ExplicitObjectRef</code>	Representa una referencia objeto explícita a un elemento. Esta clase no puede ser heredada.
<code>ListCollectionView</code>	Implementa una vista de colección para una colección basada en <code>IList</code> .
<code>ObjectDataSource</code>	Sirve como fuente de datos para un ligado de datos. Un elemento de dato Ligable puede ser especificado como tipos de lenguaje común en tiempo de ejecución.
<code>ObjectRef</code>	La clase abstracta que es usada como la clase padre de <code>ElementObjectRef</code> , <code>ExplicitObjectRef</code> , y <code>TypeObjectRef</code> .
<code>ParameterCollection</code>	Objetos de esta clase mantienen la colección de parámetros nombrados (con sus correspondientes valores) para un <code>SqlDataSource</code> .
<code>QueryCommand</code>	Esta clase representa una



120

	transacción de consulta para ser enviada ala base de datos.
RefreshCompletedEventArgs	Encapsula los argumentos pasados en el evento RefreshCompleted del ObjectDataSource, en el evento RefreshCompleted del XmlDataSource.
SqlCommandList	Una lista de comandos sql y los nombres de las tablas que estos deberían usar.
SqlDataSource	SqlDataSource toma datos de un Servidor Microsoft SQL para usarlos en el ligado de datos.
TransformerSource	Permite a los recursos referenciar a una clase de transformación que es definida como código oculto en una aplicación actual.
TypeObjectRef	Soporta referencias a objetos por tipos. Esta clase no puede ser heredada.
WinFSDataSource	El WinFSDataSource facilita el ligado de datos de datos almacenados en WinFS con aplicaciones Avalon.
XmlDataNamespaceManager	La Clase XmlDataNamespaceManager es Usada para declarar espacios de nombres para ser usados en ligados de datos Xml de consultas XPath.
XmlDataSource	Sirve como fuente de datos para datos ligados a nodos de contenido.



Lenguaje Extendido de
Etiquetas (XML).

XmlNamespace

Declara un espacio
de nombres individual
dentro de una fuente de
datos.

Interfaces

IContains

Usado para crear clases que
declaren criterio de filtrado para
vistas de colección.

IDataSource

Soporta la creación de objetos
de fuente de datos. Objetos de
fuente de datos son usados para
representaciones comunes de datos
para el ligado de datos.

IDataTransformer

Suministra métodos que permiten
la transformación en el lado del
cliente de datos ligados.

Enumeraciones

BindFlags

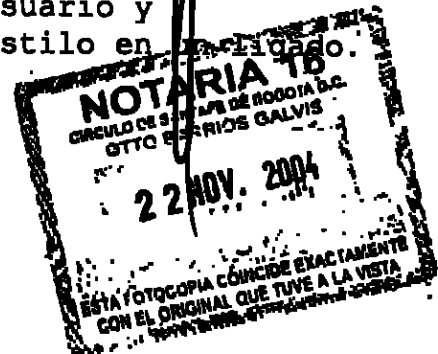
Describe propiedades
especiales de ligado. Vea el uso de
Using Bind Declarations for
"Longhorn" marca language (código
llamado "XAML"). Vea el BindType
para enumeraciones que son usadas en
un tipo específico de ligado (una
vía, dos vías y demás).

BindStatus

Estado de Ligado

BindType

Describe como los cambios en
los valores de los datos podrán ser
transferidos hacia y desde
propiedades de usuario y
propiedades de estilo en el ligado.



295

La enumeración de los modos posibles que puede tener `SqlDataSource`. El modo determina `SqlDataSourceMode` que ordenamiento de datos es retornado cuando la aplicación recupera el valor de la propiedad `Data`.

`UpdateType` Especifica cuando deben ocurrir las actualizaciones en las fuentes de datos (transferencia de datos del objetivo a la fuente) de un ligado. Asignar estos valores será relevante solo si un ligado `BindType` es asignado a un `TwoWay` (o dejado por defecto).

Delegados

`DataChangedEventHandler` Representa el método que maneja el evento `DataChanged` disparado por fuentes de datos que implementan `IDataSource`.

`DataTransferEventHandler` Representa el método que maneja el evento transferencia de datos disparado por `Binding`.

`RefreshCompletedEventHandler` Representa el método que maneja eventos `ObjectDataSource.RefreshCompleted` y `XmlDataSource.RefreshCompleted`.

Espacio de Nombres de Medios

Un espacio de nombres de medios 318 provee varias clases de medios. Los desarrolladores de aplicaciones como los desarrolladores de componentes pueden usar estas clases para desarrollar varias funcionalidades de presentación. Clases de ejemplo en el espacio de nombres



296:

de medios 318 incluyen una clase "ImageEffect" que permite ciertos efectos de imagen (por ejemplo nublar y escala de grises), y una clase "Brush" que provee mecanismos para rellenar un área usando colores sólidos, gradientes, imágenes, video y demás.

El espacio de nombres de medios 318 incluye un sub-espacio de nombres System.Windows.Media.Animation que incluye servicios que permiten al desarrollador propiedades de animación y coordinar un conjunto de animaciones con un conjunto de líneas de tiempo. Una animación es un objeto que cambia un valor sobre un periodo de tiempo. Los efectos de animación incluyen mover un objeto en la pantalla, cambiar el tamaño, la forma, o el color de un objeto. Múltiples clases de animación son proveídas para implementar varios efectos de animación. Los efectos pueden ser alcanzados asociando una animación con un valor de una propiedad de un elemento. Por ejemplo, para crear un rectángulo que se apaga dentro y fuera de una vista, una o más animaciones son asociadas con la propiedad de opacidad del rectángulo.



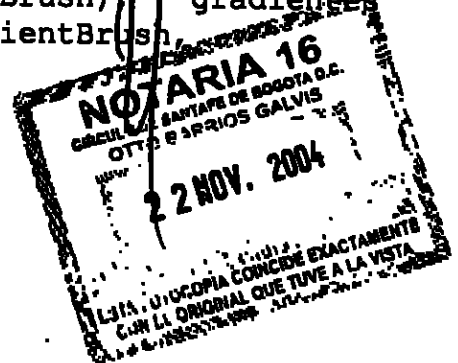
797

El espacio de nombres de medios 318 también incluye un sub-espacio de nombres System.Windows.Media.TextFormatting que suministra varios servicios de texto. Por ejemplo, un motor de texto "TextFormatter" provee servicios para rompimientos de líneas de texto y formateo de texto presentado en la pantalla. "TextFormatter" es capaz de manejar diferentes formatos de caracteres y estilos de párrafos así como manejar un esquema de texto internacional.

Las siguientes tablas listan miembros de ejemplo expuestos en el espacio de nombres System.Windows.Media.

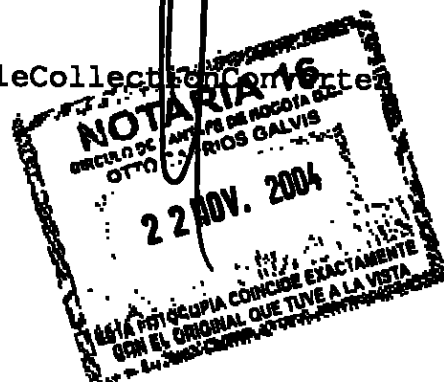
Clases

ArcSegment	Representa un arco elíptico entre dos puntos.
AudioData	Habilita para reproducir archivos de audio de acuerdo al estado de un nodo de tiempo.
AudioDataConverter	AudioDataConverter
BezierSegment	Representa una curva Bézier cúbica dibujada entre dos puntos.
Brush	Suministra un medio genérico para rellenar un área usando colores sólidos (SolidColorBrush) gradientes (LinearGradientBrush)

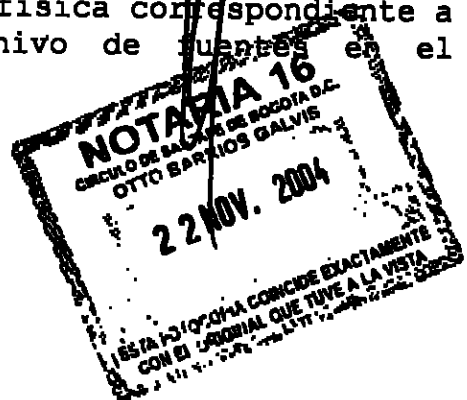


798

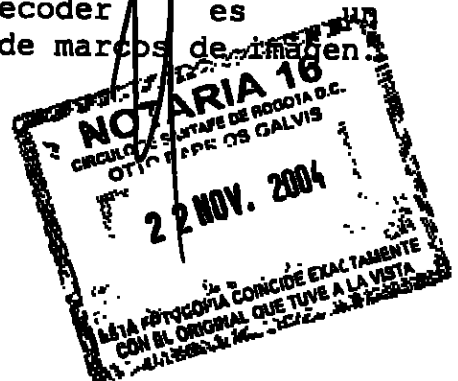
	RadialGradientBrush), imágenes (ImageBrush), video, y más.
BrushConverter	Usado para convertir un objeto Brush hacia o desde otro tipo de objeto.
Brushes	Implementa un conjunto de colores sólidos predefinidos.
CloseSegment	Representa una línea que conecta el último punto de un objeto PathFigure con su punto de inicio
CodecFilter	Filtro para enumerar codificadores. Solo aquellos codificadores que correspondan a las propiedades podrán ser enumerados.
CodecInfo	Información acerca de un codificador específico y una fábrica para crear el codificador. Este es retornado del enumerador codificador
ColorCollection	
ColorCollectionConverter	ColorCollectionConverter - Convierte clases de instancias de conversión de otros tipos hacia y desde instancias ColorCollection.
ColorContext	
ColorContext	
ColorConverter	Usado para convertir un objeto Color hacia desde otro tipo de objeto.
Colors	Implementa un conjunto de colores predefinidos
ContainerVisual	Maneja una colección de objetos Visuales.
DashArrays	DashArrays - La clase DashArrays es estática, y contiene propiedades para un mejor conocimiento de estilos mezclados.
DoubleCollection	
DoubleCollectionConverter	DoubleCollectionConverter



	- Clase Converter para convertir instancias de otros tipos hacia y desde instancias DoubleCollection.
Drawing	Un Drawing es una lista de primitivas de dibujo en dos dimensiones.
DrawingBrush	DrawingBrush - Este TileBrush define su contenido como un Drawing
DrawingContext	Contexto Drawing.
DrawingVisual	Visualización que contiene contenido gráfico para ser dibujado.
EllipseGeometry	Representa la geometría de un círculo o una elipse.
FontFamily	Familia de Fuente
FormattedText	La clase FormattedText es una parte del API de texto fácil Avalon MIL, el cual es fijado como objetivo alas necesidades de los programadores para adicionar algún texto simple en un MIL visual.
Geometry	Es una clase abstracta que suministra funcionalidad base para todas las clases de geometría, tales como EllipseGeometry, RectangleGeometry, y PathGeometry. La clase Geometry de objetos que pueden ser usados para cotar, probar, y desplegar datos gráficos en 2-D.
GeometryCollection	Representa una colección de objetos Geometry.
GetPageEventArgs	Clase GetPageEventArgs
GlyphRun	Clase GlyphRun
GlyphTypeface	La apariencia de una fuente física correspondiente a un archivo de fuentes en el disco.



GradientBrush	Una clase abstracta que describe un relleno de gradiente. Las clases que se derivan de GradientBrush describen diferentes maneras de interpretar paradas de gradiente.
GradientStop	Describe la posición y el color en un punto de transición en un gradiente.
GradientStopCollection	Representa una colección de gradientes GradientStop.
HitTestParameters	Esta es la clase base para empaquetar simultáneamente parámetros de una situación de pruebas
HitTestResult	Esta base retorna la visualización que fue exitosa durante el paso de pruebas.
HwndInterop	HwndInterop
HwndVisual	
HyphenationCandidate	Describe un candidato Hyphenation.
ICCProfile	
ImageBrush	Rellena un área con una imagen. Esta clase puede ser usada para especificar imágenes como el relleno o fondo de otros objetos.
ImageCodecCollection	La colección de codificadores (actualmente CodecInfos) en el sistema.
ImageCodecEnumerator	La enumeración de marcos de Imagen.
ImageColorTransform	ImagenColorTransform Lleva a cabo la administración del color en un ducto de imagen.
ImageData	Contiene una imagen y los datos relacionados.
ImageDataBuilder	Este objeto es usado para construir un objeto ImagenData.
ImageDecoder	ImageDecoder es un contenedor de marcos de imagen.



601

Cada marco de imagen es un ImageSource. A diferencia del ImageSource, ImageDecoder NO es un objeto incambiable y puede ser reinicializado para un flujo diferente de imágenes. Sin embargo, cualquier ImageSources (marcos) que este retorne debe ser incambiable.

ImageDecoderBmp

El Decodificador Bmp (mapa de Bips) incorporado en Microsoft.

ImageDecoderGif

El Decodificador GIF incorporado en Microsoft.

ImageDecoderIcon

El Decodificador Icon incorporado en Microsoft.

ImageDecoderInternal

Para uso interno solamente.

ImageDecoderJpeg

El Decodificador Jpeg incorporado en Microsoft.

ImageDecoderPng

El Decodificador PNG incorporado en Microsoft.

ImageDecoderTiff

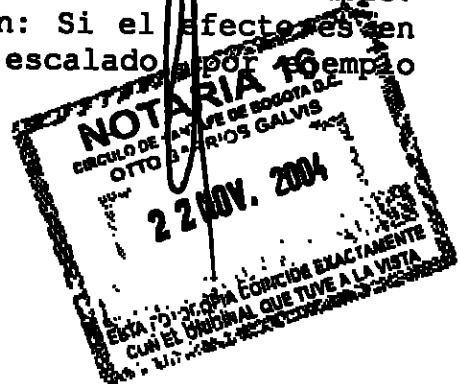
El Decodificador Tiff incorporado en Microsoft.

ImageEffect

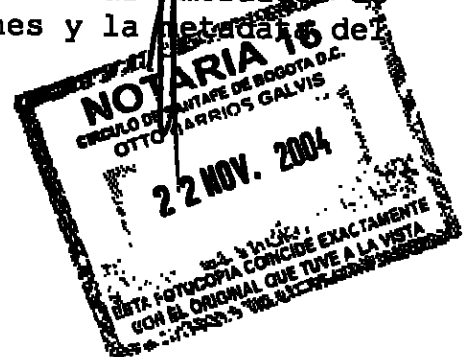
La clase ImageEffect es la clase base para todos los efectos de imágenes (oscurecer, escala de gris, etc.) Es posible que para un efecto no tenga ninguna entrada pero un efecto debe tener por lo menos una salida. La implementación por defecto de los efectos asume esto. Si un efecto derivado es reproducido con Salida/Salidas es seguro que por lo menos uno de estos estará.

ImageEffectBlur

El efecto de oscurecimiento Gaussiano. Este tiene una entrada simple, y un efecto de salida simple. Prevención: Si el efecto es en si mismo escalado por 16 veces

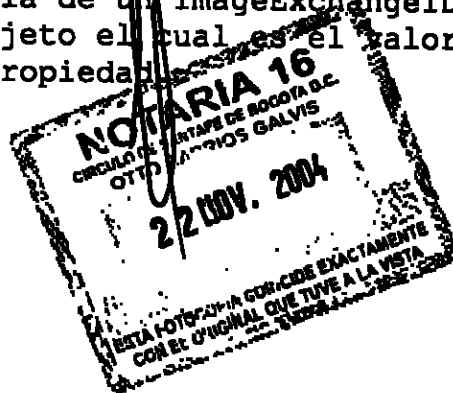


	Input.ScaleX o Input.ScaleY diferentes de 1) y Expand esta en verdadero, entonces es posible que las dimensiones de salida sean más grandes o pequeñas que el PixelWidth y el PixelHeight. Ajustar la memoria de puntos a copiar para evitar problemas.
ImageEffectFlipRotate	Este efecto puede arrojar una imagen en X o Y y rotarla en múltiplos de 90 grados.
ImageEffectGammaCorrect	Este efecto cambia el gama de una imagen.
ImageEffectGlow	Lleva a cabo un efecto de brillo. Este es un efecto de entrada sencilla y salida sencilla.
ImageEffectGrayscale	Convierte una imagen a escala de grises. Este es un efecto de entrada sencilla y salida sencilla.
ImageEffectNegate	Saca el negativo de una imagen. Este es un efecto de entrada sencilla y salida sencilla.
ImageEffectSharpen	Máscara no intensa. Este es un efecto de entrada sencilla y salida sencilla.
ImageEffectSource	Implementación de la clase ImageEffectSource
ImageEffectSourceCollection	Colección de salidas de efecto de imagen.
ImageEffectTint	El constructor Tint. Este es un efecto de entrada sencilla y salida sencilla.
ImageEncoder	ImageEncoder recolecta un conjunto de marcos (ImageSource's) con su asociado mosaico de mini-imágenes y metadatos y los guarda en un flujo especificado. Adicionalmente al mosaico de mini-imágenes y la metadata del



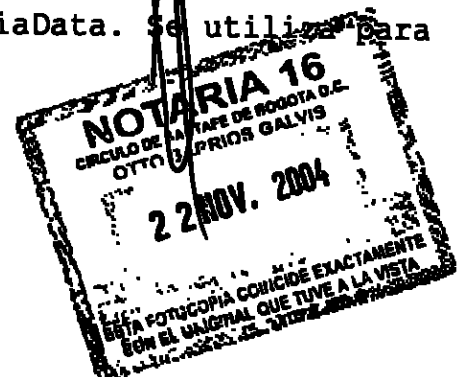
803

	marco, también puede ser una imagen extensa (global) de mini-imágenes y metadata, si el codificador lo soporta.
ImageEncoderBmp	Encoder incorporado para archivos Bmp.
ImageEncoderGif	Encoder incorporado para archivos Gif.
	ImageEncoder colecciona un conjunto de marcos (ImageSource's) con su asociado mosaico de mini-imágenes y metadatos y los guarda en un flujo determinado. Adicionalmente al mosaico de mini-imágenes y metadatos del marco, puede haber también ser una imagen extendida (global) de mini-imágenes y metadatos, si el codificador lo soporta.
ImageEncoderInternal	
	Encoder incorporado para archivos JPEG.
ImageEncoderJpeg	
	Encoder incorporado para archivos Png.
ImageEncoderPng	
	Encoder incorporado para archivos Tiff.
ImageEncoderTiff	
	La clase ImageExchangeMetaData es usada para acceder y asignar metadatos para ImageFiles los cuales tienen estilo metadata. La MetaData es almacenada en pares Clave/Valor, donde la Clave no es necesariamente única. Esta clase provee acceso genérico a toda la metadata dentro de una imagen, también expone las propiedades CLR para algunas propiedades bien conocidas.
ImageExchangeMetaData	
	ImageExchangeProperty - una tupla de un ImageExchangeID y el objeto el cual es el valor de la propiedad.
ImageExchangeProperty	

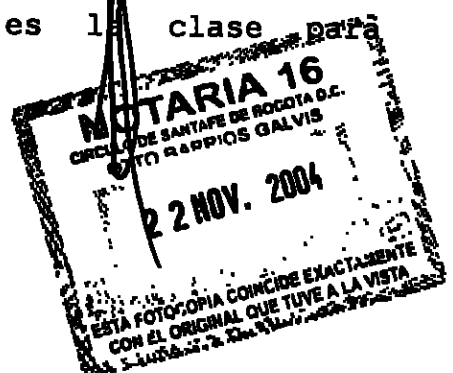


804

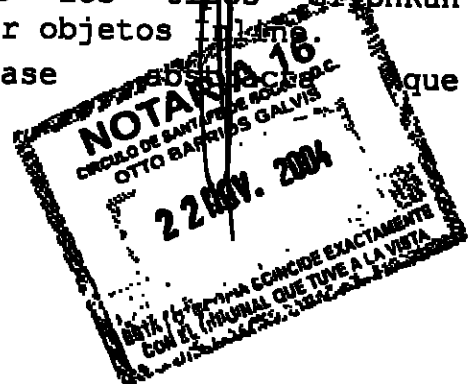
ImageMetaData	La clase ImageMetaData es usada para acceder un conjunto de metadatos de Imágenes. Esta clase también expone una propiedad CodecMetaData la cual expone un decodificador-específico accediendo a la metadata de la imagen.
ImagePalette	Clase ImagePalette
ImageSizeOptions	Opciones de tamaño para una imagen. La imagen resultado será escalada con base en estas opciones.
ImageSource	Define los métodos, propiedades, y efectos para el conducto de la imagen, incluyendo decodificadores y efectos.
ImageSourceCollection	La colección de codificadores (actualmente ImageSource's) en el sistema.
ImageSourceConverter	ImageSourceConverter
IntegerCollection	IntegerCollectionConverter
IntegerCollectionConverter	- Clase Converter para convertir instancias de otros tipos hacia y desde instancias IntegerCollection.
LinearGradientBrush	Define un gradiente lineal usado para rellenar un área.
LineGeometry	Representa la geometría de una línea.
LineSegment	Representa una línea entre dos puntos. Diferente a los objetos LineGeometry, LineSegment podría estar contenido dentro de un PathFigure.
MatrixTransform	Crea una transformación arbitraria de matriz afín usada para manipular sistemas de objetos o coordenadas en un plano de dos dimensiones.
MediaData	MediaData. Se utiliza para



	retroceder contenido de Audio/Video.
MediaSystem	La clase MediaSystem controla la capa de medios.
NineGridBrush	Rellena un área entera con una imagen. Partes de la imagen son ajustadas dentro de las márgenes definidas.
PathFigure	Representa una sub-sección de una geometría, una serie conectada simple de segmentación métricos en dos dimensiones.
PathFigureCollection	
PathFigureConverter	PathFigureConverter
PathGeometry	Representa una forma compleja que puede ser compuesta de arcos, curvas, elipses, líneas, y rectángulos.
PathGeometryConverter	PathGeometryConverter
	Es una clase abstracta que representa un segmento del objeto PathFigure. La clases que deriven de PathSegment, tales como ArcSegment, BezierSegment, y LineSegment, representan tipos específicos de segmentos geométricos.
PathSegment	Representa una lista de objetos PathSegment.
PathSegmentCollection	PathSegmentConverter
PathSegmentConverter	Describe como una forma es perfilada.
Pen	PixelFormats - La colección de Formatos Pixel soportados.
PixelFormats	
PointCollection	PointCollectionConverter - Clase Converter para convertir instancias de otros tipos hacia y desde instancias PointCollection.
PointCollectionConverter	Esta es la clase para
PointHitTestParameters	



	especificar los parámetros de pruebas en un punto.
PointHitTestResult	Esta clase retorna el punto y la visualización del paso de las pruebas.
PolyBezierSegment	PolyBezierSegment
PolyLineSegment	PolyLineSegment
PolyQuadraticBezierSegment	PolyQuadraticBezierSegment
PrintContext	PrintContext mantiene un estado y un contexto para una interacción con la impresora.
QuadraticBezierSegment	QuadraticBezierSegment
	Define un gradiente radial usado para rellenar un objeto. Un punto focal define el comienzo del gradiente, y un círculo define el punto final del gradiente.
RadialGradientBrush	
	Representa la geometría de un rectángulo
RectangleGeometry	
RetainedVisual	RetainedVisual
	Usado para rotar un objeto alrededor de un punto especificado en un plano de dos dimensiones x-y.
RotateTransform	
	Escala un objeto en el plano de dos dimensiones x-y, comenzando en un punto central definido. Los factores de escala son definidos en las direcciones X y Y de este punto central.
ScaleTransform	
	Representa un desvío en dos dimensiones.
SkewTransform	
SolidColorBrush	Representa un relleno sólido, uniforme.
StartSegment	StartSegment
	Colección de sublíneas. Sublínea puede ser un objeto de uno de los tipos GlyphRun LineOver objetos
SubLineCollection	
TileBrush	Clase que



801

describe una manera de rellenar una región con uno o más "loza." Las clases derivadas definen los diferentes tipos de lozas que pueden ser usados; por ejemplo, el ImageBrush lo habilita para rellenar un área con una imagen.

Transform

Una clase abstracta que usted puede usar como la clase padre de todos los tipos de transformaciones en el plano de dos dimensiones, incluyendo rotación (RotateTransform), escalamiento (ScaleTransform), desvío (SkewTransform), y traslación

(TranslateTransform). Esta jerarquía de clases difiere de la estructura Matrix debido a que esta es una clase y esta soporta animación y semántica de enumeración.

TransformCollection

Usado para crear y manipular una lista de objetos Transform.

TransformConverter

Usado para convertir un objeto Transform hacia y desde otro tipo de objeto.

TranslateTransform

Transforma un objeto en el plano bidimensional x-y.

Typeface

Un Typeface es una combinación de familias, peso, estilo y extensión:

VectorCollection

VectorCollectionConverter

VectorCollectionConverter

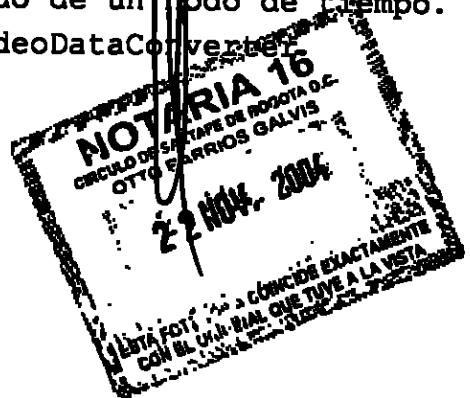
- La clase Converter para convertir instancias de otros tipos hacia y desde instancias VectorCollection.

VideoData

Habilita la reproducción de archivos de video de acuerdo al estado de un modo de tiempo.

VideoDataConverter

VideoDataConverter



868

Visual	Clase base para todos los tipos de Visualización. Esta suministra servicios y propiedades comunes a todas las visualizaciones, incluyendo pruebas de choque, transformación de coordenada, y limitación en la caja de cálculos.
VisualCollection	Una colección ordenada de objetos Visual.
VisualManager	Representa un árbol de objetos Visuales para un destino de representación, típicamente una ventana.

Interfaces

IHyphenate	IHyphenate es la Interfaz para el proveedor de servicio de concatenación.
IRetainedRender	Si ésta Interfaz es implementada en una clase que es derivada de RetainedVisual, la operación RetainedVisual en modo de validación, por ejemplo, el sub-sistema gráfico podrá ser un llamado a OnRender de forma lenta. (Por ejemplo si la Visualización aparece por primera vez en la pantalla). Nótese que OnRender puede ser llamado por el sistema a cualquier momento.
IVisual	Esta Interfaz define los métodos comunes y servicios disponibles de un objeto Visual.

Enumeraciones



809

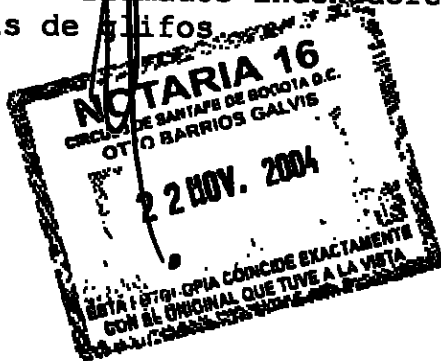
BrushMappingMode	BrushMappingMode - Enumeración que describe cuando ciertos valores deberían ser considerados como coordenadas locales absolutas o cuando ellas deberían ser consideradas múltiplos de un tamaño limitado por una caja.
ChannelDescription	Describe el orden de cada canal de dato de píxel
ColorInterpolationMode	ColorInterpolationMode - Este determina como los colores en un gradiente son interpolados.
CombineMode	Especifica el método usado para combinar dos áreas geométricas.
FillRule	
GradientSpreadMethod	Especifica como el gradiente debe ser dibujado fuera del vector gradiente especificado o espacio.
HitTestFilterBehavior	Comportamiento de visualizaciones filtradas mientras son probadas.
HitTestResultBehavior	Enumera comportamientos de control cuando una prueba positiva ocurre durante el probado.
HorizontalAlignment	La enumeración HorizontalAlignment es usada para describir como el contenido es posicionado horizontalmente dentro de un contenedor.
HyphenationRule	Soporta reglas Hyphenation.
ImagePaletteType	Tipos de paletas pre-definidas.
MediaState	Mantiene el estado actual del Medio.
PenDashCap	PenDashCap - Enumeración la cual describe las líneas finales de un trazo dentro de



	una línea trazada.
PenLineCap	Describe la forma en el final de una línea o segmento.
PenLineJoin	PenLineJoin - Enumeración la cual describe el dibujo de las esquinas en una línea.
Rotation	La rotación a ser aplicada; solo son soportados múltiplos de 90 grados.
StandardColorSpace	
Stretch	Stretch - Enumeración la cual describe como la fuente de una recta debe ser extendida para fijarse a una recta destino.
StyleSimulations	Fuente simulación de estilo
TiffCompressOptions	Opciones de compresión para guardar una imagen TIFF.
TileMode	TileMode - Enumeración la cual describe el dibujo en los finales de una línea.
VerticalAlignment	La enumeración VerticalAlignment es usada para describir como el contenido es posicionado verticalmente dentro de un contenedor.

Estructuras

CharacterIndexer	Esta clase es un ayudante para implementar los llamados Indexadores para caracteres.
Color	Representa olores en términos de canales alfa, rojo, verde, y azul.
GlyphIndexer	Esta clase es una ayudante para implementar los llamados indexadores para métricas de glifos.



B11

ImageExchangeID

ImageExchangeID - Esta clase es el tipo el cual puede ser usado como una llave para una propiedad en una instancia de ImageMetaData. Esta puede ser entero o caracter..

ImageExchangeMetaDataEnumerator

ImageExchangeMetaDataEnumerator Es el enumerador para ImageExchangeMetaData. Contiene una Interfaz IEnumerator como versiones de tipo fuertes de los APIs

ImageFrameEnumerator

El enumerador de marcos de Imagen.

ImageMetaDataRational

Una clase ImageMetaDataRational es representada como un numerador con signo y denominador con signo. El valor efectivo de un racional es el numerador/denominador.

ImageMetaDataUnsignedRational

Una clase racional es representada como un numerador sin signo y n denominador sin signo. El valor efectivo de un racional es el numerador/denominador.

ImagePaletteColor

Estructura ImagePaletteColor.

IntegerRect

Una recta compuesta de valores enteros. Típicamente usada para especificar la recta fuente (en puntos) de interés en una imagen.

Matrix

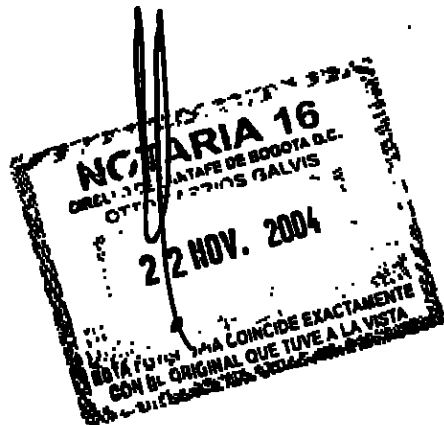
Representa una matriz de 3x3 usada para transformaciones en el espacio de dos-dimensiones. Por lo que "Avalon" solo permite transformaciones afines, la estructura Matrix tiene seis entradas en lugar de nueve..

NamedStringIndexer

Esa clase es un ayudante para implementar el llamado indexador de cadenas localizado en múltiples culturas.

PixelFormat

Definición de Formato de Puntos para imágenes y superficies basadas en puntos.



810
--
812

Delegados

GetPageEventHandler	Delegado GetPageEventHandler
HitTestFilterDelegate	Delegado para probar aciertos para controlar cuando provar contra un hijo de una visualización
HitTestResultDelegate	Delegado para probar aciertos para controlar el retorno de una información exitosa en una visualización.

La siguiente lista de tablas de miembros ejemplo
expuestos por el espacio de nombres
System.Windows.Media.Animation namespace.

Classes

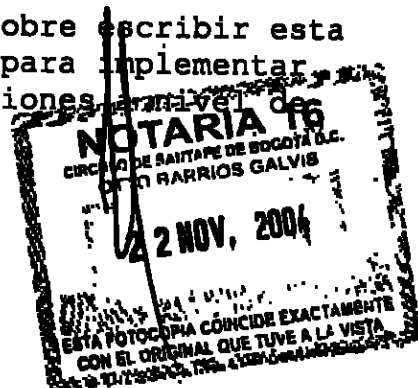
Animatable

Cualquier clase que no deriva de DependencyObject pero la cual tiene propiedades que pueden ser animadas debería derivar de esta clase.

AnimationCollection	Esta clase abstracta suministra funcionalidad base para colecciones de animaciones, tales como ColorAnimationCollection , DoubleAnimationCollection , y SizeAnimationCollection .

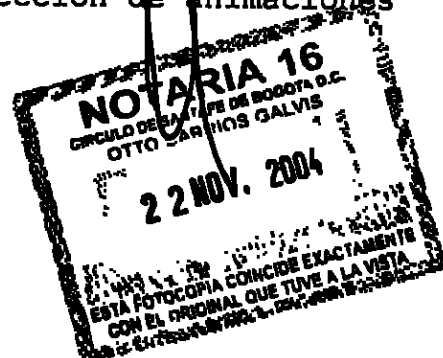
AnimationEffect

Sobre escribir esta
clase para implementar
animaciones a nivel de

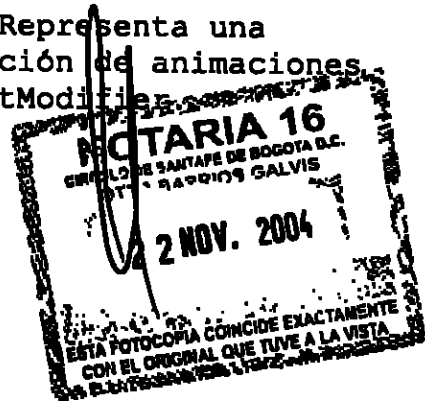


22

	elemento las cuales pueden participar en el proceso de representación para instanciar animaciones en múltiples elementos en el tiempo de representación.
AnimationEffectCollection	Mantiene una colección de AnimationEffects.
BoolAnimationCollection	Representa una colección de animaciones BoolModifier animations.
BoolModifier	
BoolTimedModifier	
ByteAnimationCollection	Representa una colección de animaciones BoolModifier.
ByteModifier	
ByteTimedModifier	
CharAnimationCollection	Representa una colección de animaciones CharModifier.
CharModifier	
CharTimedModifier	
ColorAnimation	Anima un valor de color de una propiedad.
ColorAnimationCollection	Representa una colección de animaciones ColorModifier animations.
ColorKeyFrameCollection	
ColorModifier	
ColorTimedModifier	
DecimalAnimationCollection	Representa una colección de animaciones DecimalModifier.
DecimalModifier	
DecimalTimedModifier	
DoubleAnimation	Usado para animar propiedades que aceptan un valor doble.
DoubleAnimationCollection	Representa una colección de animaciones



DoubleKeyFrameCollection	DoubleModifier animations.
DoubleModifier	
DoubleTimedModifier	
FloatAnimation	Usado para animar propiedades que aceptan un valor simple.
FloatAnimationCollection	Representa una colección de animaciones FloatModifier animations.
FloatKeyFrameCollection	
FloatModifier	
FloatTimedModifier	
IntAnimationCollection	Representa una colección de animaciones IntModifier animations.
IntModifier	
IntTimedModifier	
LengthAnimation	Usado para animar propiedades que aceptan un valor Length.
LengthAnimationCollection	Representa una colección de animaciones LengthModifier animations.
LengthKeyFrameCollection	
LengthModifier	
LengthTimedModifier	
LongAnimationCollection	Representa una colección de animaciones LongModifier animations.
LongModifier	
LongTimedModifier	
MatrixAnimationCollection	Representa colecciones de animaciones MatrixModifier.
MatrixModifier	
MatrixTimedModifier	
Modifier	
ObjectAnimationCollection	Representa una colección de animaciones ObjectModifier animations.



ObjectModifier

ObjectTimedModifier

PathAnimation

Esta animación puede ser usada dentro de un **MatrixAnimationCollection** para mover un objeto visual a lo largo de una ruta.

PointAnimation

Usado para animar propiedades que acepten valores **Point**.

PointAnimationCollection

Representa una colección de animaciones **PointModifier**.

PointKeyFrameCollection

PointModifier

PointTimedModifier

RectAnimation

Usado para animar propiedades que acepten un valor **Rect**.

RectAnimationCollection

Representa una colección de animaciones **RectModifier** animations.

RectKeyFrameCollection

RectModifier

RectTimedModifier

ShortAnimationCollection

Representa una colección de animaciones **ShortModifier**.

ShortModifier

ShortTimedModifier

SizeAnimation

Define una animación basada en el tamaño de un objeto. Suministra información de tamaño, un objeto puede aparecer como encogido o alargado en un periodo de tiempo.

SizeAnimationCollection

Representa una colección de animaciones **SizeModifier**.

SizeKeyFrameCollection



SizeModifier

SizeTimedModifier

StringAnimationCollection Representa una colección de animaciones StringModifier.

StringModifier

StringTimedModifier

Timeline Mantiene el estado de tiempo en tiempo de ejecución para objeto de tiempo.

TimelineBuilder Un objeto que puede ser usado para crear objetos Timeline.

TimeManager El objeto que controla un árbol de tiempos completo.

TimeSyncValueTypeConverter Un objeto que lleva a cabo tipos de conversiones que involucran valores TimeSyncValue.

TimeTypeConverter Un objeto que lleva a cabo tipos de conversiones que involucran valores Time.

VectorAnimation Usado para animar propiedades que aceptan valores Vector.

VectorAnimationCollection Representa una colección de animaciones VectorModifier.

VectorKeyFrameCollection

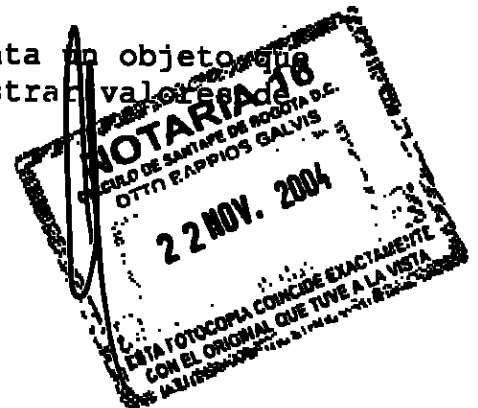
VectorModifier

VectorTimedModifier

Interfaces

IClock

Representa un objeto que puede suministrar valores de tiempo.



817

tiempo lineal.

IModifier

Define el comportamiento básico de un objeto modifier. Un modificador es un objeto que toma otro objeto, llamado el *valor base*, cierto tipo y retorna otro objeto del mismo tipo como una salida.

ITimingControl

Define el comportamiento de líneas de tiempo y objetos de tiempo.

ITimingControlBuilder

Representa un objeto que puede construir una plantilla de línea de tiempo.

Enumeraciones

AnimationType

Describe el comportamiento de una animación.

CloneType

Los tipos de clones que CloneCore puede solicitar.

InterpolationMethod

Describe como una animación calcula sus valores de salida.

KeyTimeType

Los diferentes tipos de KeyTimes.

TimeEndSync

Valores para el atributo endSync, los cuales especifican como un contenedor calcula su simple duración basado en la duración de los hijos.

TimeFill

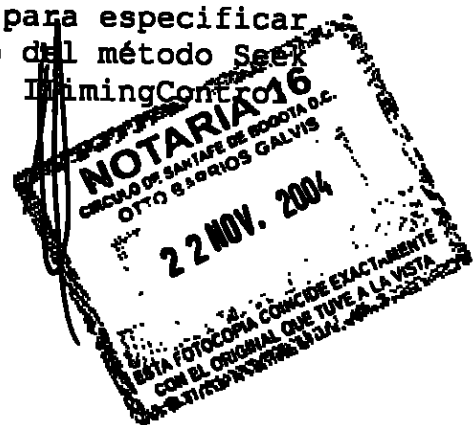
Específica como las líneas de tiempo proceden después de que estas no tienen no tienen actividad prolongada.

TimeRestart

Valor del atributo Timeline.Restart.

TimeSeekOrigin

Indica una posición de línea de tiempo; usada para especificar el comportamiento del método Seek de las Interfaces ITimingControl



definiendo la posición en la cual el desplazamiento es aplicado.

TimeSyncBase

El evento para sincronizar un valor de inicio o de fin.

Estructuras

ColorKeyFrame

DoubleKeyFrame

FloatKeyFrame

KeySpline

Esta clase es usada para pasar un arreglo de splines llave a la propiedad KeySplines de un fragmento de animación.

KeyTime

Un KeyTime es usado para especificar cuando se hace referencia al tiempo de una animación donde KeyFrame toma lugar.

LengthKeyFrame

PointKeyFrame

RectKeyFrame

SizeKeyFrame

Time

Un valor representando tiempo, con operaciones aritméticas de tiempo asociadas.

TimelineEnumerator

Enumera elementos en una colección TimelineList.

TimeSyncValue

Un valor representando un tiempo de comienzo o de fin absoluto o relativo para una línea de tiempo.

VectorKeyFrame



Las siguientes tablas listan miembros de ejemplo expuestos por el espacio de nombres System.Windows.Media.TextFormatting namespace.

Clases

InlineObjectInfo

Suministra detalles de medida para objetos de texto en línea. El cliente formateado pasa este objeto como un parámetro al método GetInlineObjectInfo.

TextFormatter

TextFormatter es un motor de texto "Avalon" y suministra servicios para formatear textos y hacer rompimiento en líneas de texto. Los TextFormatter pueden manejar diferentes formatos de caracteres de texto y estilos de párrafos, e incluye soporte para esquemas de texto internacional.

TextHighlightBounds

Liga un rango de texto.

TextInfo

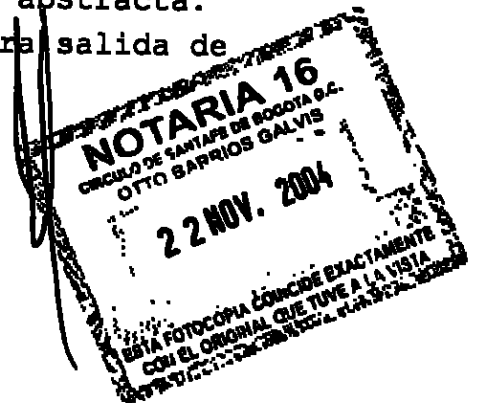
Representa información acerca de un bloque de texto en el almacenamiento de caracteres fuente de texto en el cliente.

TextLine

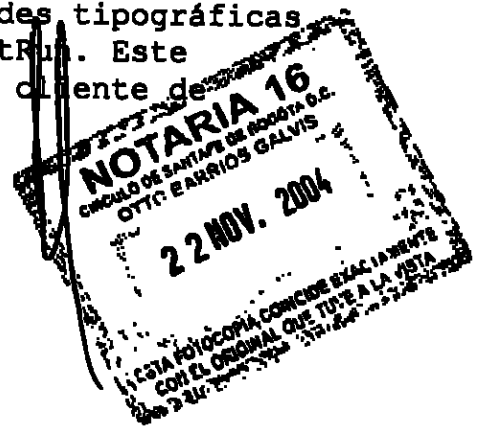
Suministra servicios a una línea de texto. Herede de esta clase para implementar servicios que manipulen y formateen una línea de texto. Esta es una clase abstracta.

TextMarkerGeneratedContent

Genera salida de



	marcador de lista de línea. Define el estilo y el tipo de un marcador de lista de párrafos. El formateo de cliente usa esta clase como parámetro para suministrar detalles de marcado al método <code>GetTextMarkerInfo</code>. Representa propiedades que pueden cambiar de un párrafo al siguiente, tales como dirección del flujo como, alineación, o indentación. Define una secuencia de caracteres que comparten un conjunto simple de propiedades. El cliente formateado provee detalles de <code>TextRun</code> en esta clase cuando el <code>TextFormatter</code> pasa esta como un parámetro al método <code>GetTextRun</code>. Ligado de un texto corrido. Suministra servicios de cache al objeto <code>TextFormatter</code> para mejorar su rendimiento. Representa datos de información de cliente asociados con un <code>TextRun</code>. Suministra propiedades que pueden cambiar de un <code>TextRun</code> a otro, tales como tipo o forma de brocha. Esta es una clase abstracta. Suministra propiedades tipográficas para <code>TextRun</code>. Este conjunto de cliente de
<code>TextMarkerInfo</code>	
<code>TextParagraphProperties</code>	
<code>TextRun</code>	
<code>TextRunBounds</code>	
<code>TextRunCache</code>	
<code>TextRunClientData</code>	
<code>TextRunProperties</code>	
<code>TextRunTypographyProperties</code>	



propiedades genera un conjunto de características que son procesadas por el motor de esquema OpenType.

TextSource

Suministra datos caracteres y propiedades formateadas para el TextFormatter. Todo el acceso al texto en el TextSource es alcanzado a través del método GetTextRun, el cual es designado para permitir al cliente visualizar el texto en alguna forma escogida. Esta es una clase abstracta.

TextTrimmingInfo

Suministra descripción de características de adaptación de texto. El cliente de formateo completa los detalles de adaptación en esta clase cuando el TextFormatter la pasa como parámetro del método GetTextTrimmingInfo.

Enumeraciones

TextParagraphFlags	Banderas que describen características de un párrafo.
TextRunCacheFlags	Tipo de contenido en el cache de texto corrido.
TextRunType	Indica el tipo de TextRun.



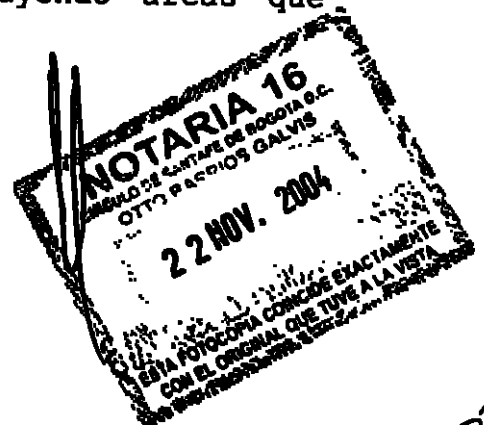
Estructuras

TextSourceCharacterIndex
TextSourceCharacterIndex representa una posición de carácter en el texto.

Capa de Integración de Medios

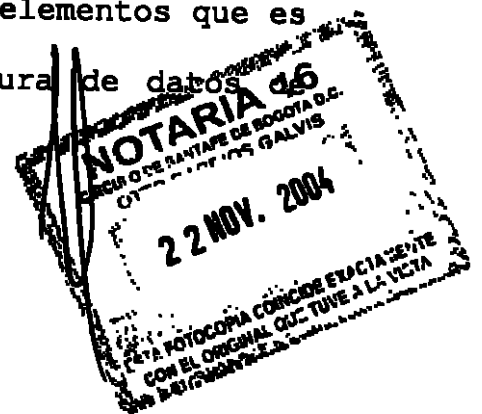
La capa de integración de medios (MIL) suministra un API para ver desarrolladores o programadores para lograr posibles efectos de composición complejos dentro de sus aplicaciones de manera correcta, mientras influencia la unidad de procesamiento de gráficas de manera que no impacte negativamente el rendimiento normal de la aplicación. Un aspecto de MIL suministra la habilidad de combinar diferentes tipos de medios (por ejemplo 2D, 3D, Video, Audio, texto, imágenes y demás) y los anime con facilidad y suavidad.

El MIL suministra una arquitectura gráfica para una composición multi-estado y un modelo de programación que permita una paridad funcional en Interfaces programáticas y descritas. Un API y un script permiten la creación de una estructura retenida o descripción de escena que es compuesta cuando es representada, incluyendo áreas que tienen naturaleza de modo inmediato.



Por medio de Interfaces, el MIL suministra acceso a una estructura de datos para almacenar información visual tal que las aplicaciones puedan tomar ventaja de las capacidades gráficas proveídas por el hardware del computador. Las Interfaces son basadas en un modelo de objeto de elementos y en un lenguaje de aumento de vectores gráficos para usar el modelo de objeto de elemento de una manera que permita programar código desarrollado a una Interfaz consistente con una estructura de datos gráfica para producir gráficas. La estructura de datos también puede ser usada para representar directamente o para "compilar" la información visual tal que esta pueda suministrar un sistema de gráficos de bajo nivel para composiciones rápidas y animaciones.

El modelo de objetos de elementos de vectores gráficos generalmente corresponde a elementos modelados y otros elementos incluyendo imagen y video que se correlacionan con un modelo de objetos de vista gráfica del escenario gráfico. El aumento puede ser analizado en datos e incluye elementos en un árbol de elementos que es traducido en los objetos de una estructura de datos



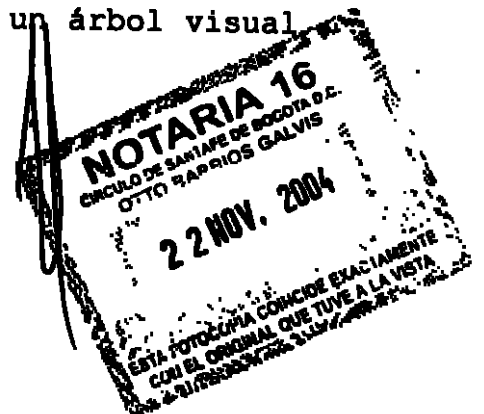
visualización gráfica. Otra ganancia puede ser trasladada directamente en los datos y llamados que crean el escenario de objetos gráficos. El lenguaje mejorado suministra distintas maneras de escribir un elemento, incluyendo un formato simple de cadena o sintaxis de propiedades complejas, las cuales pueden ser nombradas, habilitando el reuso de otras posiciones en la mejora.

Un aspecto de MIL es la integración de animaciones y tiempos a través del conjunto API, suministrando información como un concepto inherente a nivel base. Para facilitar una animación fácil, el MIL suministra un sistema de procesamiento gráfico multinivel y un método (por ejemplo un sistema operacional). Un sistema semejante de procesamiento de gráficos de múltiple nivel comprende dos componentes, incluyendo una señal por demanda o un componente de alto nivel de baja señal, como un componente de bajo nivel de señal rápida (por ejemplo en las ratas de refresco del marco de hardware gráfico). En general, el alto nivel, una componente de menor frecuencia lleva a cabo aspectos computacionalmente intensivos de actualización de parámetros de animación y atravesando estructuras de parámetros y estructuras de datos, para pasar estructuras de datos simplificados.



componente de bajo nivel. La componente de bajo nivel opera a alta frecuencia, tal como la rata de refresco de marcos de n subsistema gráfico, para procesar estructuras de datos en una salida de datos constante para el subsistema gráfico. El procesamiento de bajo nivel incluye interpolación de cualquier intervalo de parámetro como sea necesario para obtener valores instantáneos para representar el escenario de cada marco de la animación.

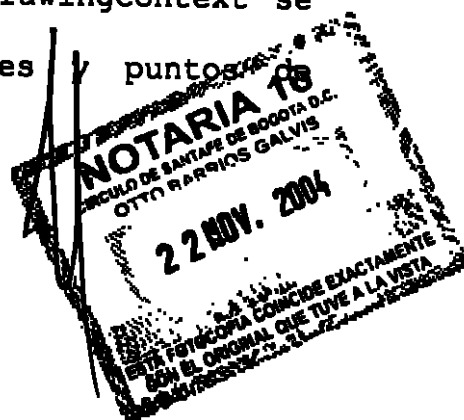
Los objetos de alto nivel MIL incluyen un árbol visual, el cual es un objeto que contiene el contenido principal del dibujo. Los controles se derivarán directamente de las visualizaciones del árbol. Las visualizaciones son dispositivos y contextos padre independiente. El objetivo de despliegue es el dispositivo en el cual el dibujo es visualizado. Este objeto (pantalla) puede tener su propio mecanismo de invalidación. Varios objetivos de despliegue pueden ser una pantalla en una ventana, una impresora, un meta archivo, una superficie y una "sub-ventana" la cual hace parte de la escena que se está dibujando separadamente en el resto de la escena. Otros objetos relacionados de dibujo incluyen un desplegador visual, que comprende de un objeto que es configurado para dibujar un árbol visual



en el objetivo de despliegue, un objeto para agendar la pantalla que conoce cuando dibujar el árbol visual en el objetivo de despliegue. Un administrador de tiempo es un objeto contexto para asignar nodos de tiempo, y es el objeto que agenda señales de llamado.

Se suministra de un API visual el cual es esencialmente el punto de inicio para dibujar por la vía de la capa de integración de medios, y comprende de múltiples tipos de objetos, incluyendo el objeto VisualManager, el cual conecta un árbol visual a un medio. Los diferentes tipos de VisualManagers (por ejemplo., Pantalla, impresora, superficie) son responsables del despliegue del árbol visual en un medio particular. Una visualización es donde el programador hace el dibujo; este es un nodo en el árbol de visualización y suministra un lugar para un programa de dibujo.

El API DrawingContext presenta un modelo de programación basado en contextos para saber de que modo construir un contenido visual que poble una visualización o despliegue un ImageData. Las clases DrawingContext se suministran así como también las clases y puntos

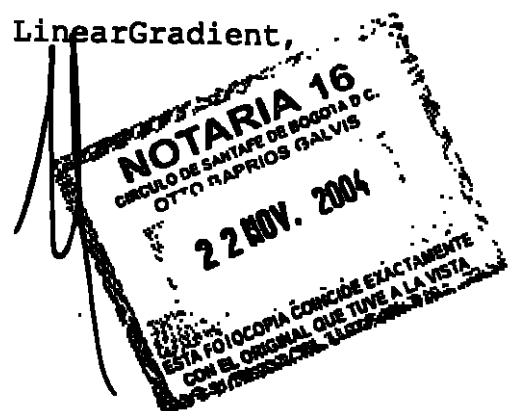


827

entrada necesarios para adquirir un `DrawingContext` y enumerar el contenido visual en un `RetainedVisual/DrawingVisual`.

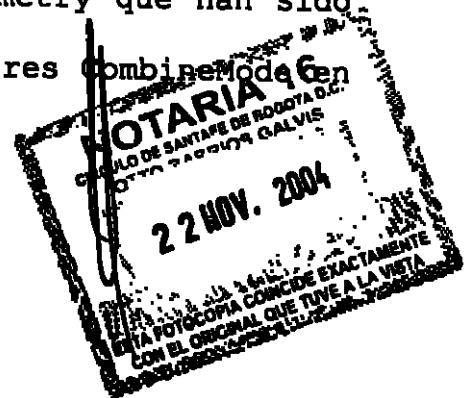
Para permitir mutabilidad, se suministra un conjunto único de tipos que derivan de una clase base común `Changeable`. Cualquier tipo para el cual la mutabilidad se desea cambiar puede derivar de la clase `Changeable`. En programación de gráficos, el modelo de objeto incluye `Brushes`, `Pens`, `Geometries`, `FloatAnimations`, `GradientStops`, `Segments`, y demás. Una propiedad `IsChangeable` especifica si el objeto cambiante puede ser modificado o no, dependiendo de su valor actual, el cual define un estado.

Una brocha es un objeto que representa un método para rellenar un plano. Adicionalmente tener la habilidad de rellenar un plano en una forma absoluta, las brochas de la capa de integración de medios están también en capacidad de adaptar como rellenar el plano relativo al tamaño del objeto que se está rellenando. Ejemplos de tipos de brochas incluyen `SolidColorBrush`, `VisualBrush` (Elcualpuede referenciar una gráfica de vector recurso/visual), `DrawingBrush`, `LinearGradient`,



RadialGradient, ImageBrush y NineGridBrush. Ciertos objetos de brocha tendrán una idea de cómo ellos se relacionan con el sistema de coordenadas cuando estos son usados, y una idea de cómo ellos se relacionan con a la caja limite de la geometría que se está usando. Este tamaño es basado en los objetos que la brocha está rellenando. La clase base Brush tiene un Transform, una opacidad general, y u modo de combinación. Los objetos Brush (y otros objetos de recurso en Vector Graphics y en el API MIL) son cambiables y son escribibles después de que ellos han sido creados, y siguen un patrón general cambiable para el modo en que ellos se comportan después de que son usados calificadamente.

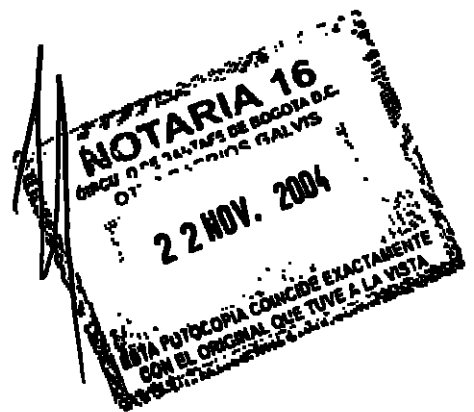
La clase Geometry de objetos puede ser usada fijando, probando y desplegando datos basados en vectores 2D con el Pen y el Brush. La clase derivada Geometry suministra semánticas específicas de construcción y enumeración. Un número de tipos Geometry es suministrado, así como un PathGeometry que permite la definición explícita de formas más complejas de Geometry. Geometry es una clase base abstracta. Una GeometryCollection es una colección de múltiples objetos Geometry que han sido combinados usando operaciones particulares



su área definida. Este objeto permite construir fácilmente combinaciones visuales de objeto Geometry que hacen posible usar estrictamente objetos PathFigure dentro de un PathGeometry.

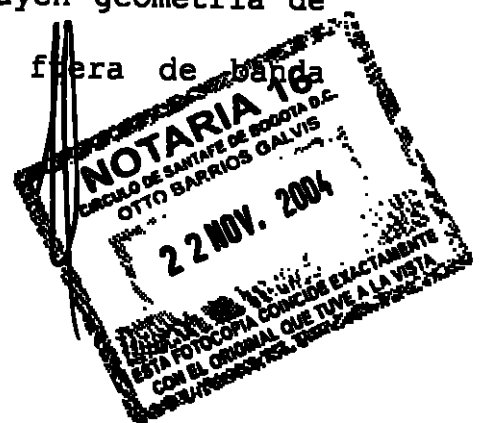
ImageSource es una clase abstracta, que comprende un bloque de construcción básica para visualizar. Un ImageSource conceptualmente representa un único, constante conjunto de puntos de un cierto tamaño y resolución. Por ejemplo, un ImageSource puede ser un único frame en un archivo de imágenes que un Decoder puede suministrar, o este puede ser el resultado de una transformación que opera un determinado ImageSource de su propiedad. Un ImageSource es cambiable, no a causa de que sus propiedades puedan ser cambiadas, sino por que las propiedades de sus subclases pueden ser potencialmente cambiadas.

La clase Transform de objetos es suministrada para escalar, rotar, trasladar y desvía vectores y gráficos. La clase derivada Transform suministra semánticas de enumeración y usos amigables.



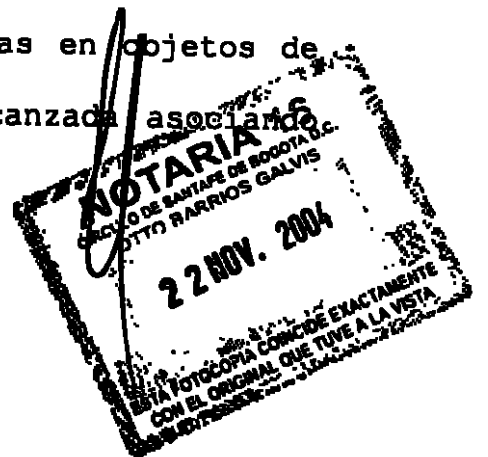
Los efectos suministran un medio de alterar el contenido visual de una escena a manera de representación centrada. Por ejemplo, los VisualEffects (en mapas de bits basados en raster) operan basados en la imagen, componiendo completamente la representación de una parte de la escena. Los efectos sub divididos en varios tipos incluyendo VisualEffects, BlendModes y VectorEffects. VisualEffects pueden ser usados en una escena de modo retenido aplicandolos a un subgrafo a sub-grafo, a un Element, o estos pueden ser usados en proceso automático de imagen. BlendModes son efectos específicos basados en imagen, y pueden ser aplicados en modo retenido de escena de igual manera que los VisualEffects. El modo Blend lleva a cabo una combinación de colores fuente y destino de acuerdo a como se compone la fuente, por ejemplo multiplicando o adicionando.

Las pruebas desiertas se usan para escoger visualizaciones en una escena, y funcionan comenzando desde el tope del árbol de control, y retornan un control o un conjunto de controles por un punto o una geometría. Un control puede definir cuando es un acierto o cuando no lo es con servicios de soporte que incluyen geometría de representación, caja límite, geometría fuera de banda



(región de acierto), opacidad de imagen o máscara, y su propia lógica. El control puede retornar un acierto específico relacionado con datos de acierto. El mecanismo de pruebas de acierto puede filtrar los resultados de las pruebas de una manera eficiente. Pruebas de acierto recorren de izquierda a derecha el árbol de visualización, con su reporte de éxitos a través de rellamados en orden, de incio a fin. Cuando descienden, vistas de prueba filtran en términos de relaciones a nivel de elemento, por ejemplo, una página lógica con formas, o un panel con una página lógica interna. Cuando ocurre un éxito, el probador puede continuar procesando más éxitos (si hay), o detenerse.

Un sistema de animación es suministrado, y comprende de un motor de control de tiempo y un conjunto de objetos de animación. El motor de tiempo es un servicio que puede ser usado por cualquier objeto que muestre comportamientos de variación de tiempo, por ejemplo animaciones y objetos de medios de video o audio. Los objetos de animación implementan un conjunto de funciones que mapean extensiones de tiempo en otros tipos de datos, los cuales luego son usados como entradas en objetos de alto nivel. La animación gráfica es alcanzada asociando

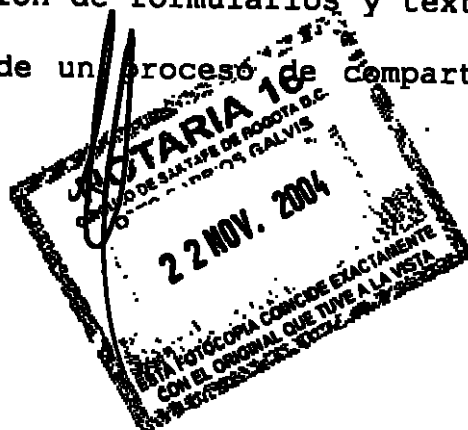


una colección de animación con la operación de despliegue. Cada animación usada en una operación de despliegue puede correr con un reloj separado, refiriéndose a la llamada "línea de tiempo". Lo que fue una animación primitiva es dibujado y especificado en los parámetros de animación, el sistema de representación de bajo nivel se preocupa de redibujar la escena a intervalos regulares. Cada marco de tiempo es representado y el valor actual de las animaciones involucradas en la escena son calculadas, basadas en el tiempo transcurrido (en muchos casos medido por el reloj de sistema), y luego las primitivas de animación son redibujadas.

Varios tipos primitivos, características de color y soporte a medios es también suministrado por medio del MIL. MediaData puede ser utilizado para ejecutar contenido de audio/video.

Espacio de Nombres de Diseño

El espacio de nombres de diseño 320 suministra clases que posibilitan la edición de formularios y texto, formateando datos y a través de un proceso de compartir



833

de datos. Estas clases suministran una estructura extensible para evitar documentos, aplicaciones, y otro contenido.

El espacio de nombres de diseño 320 contiene funcionalidad en dos niveles: funcionalidad de alto nivel para desarrolladores de aplicaciones que desean usar editores pre empaquetados para diferentes tipos de información; una funcionalidad de bajo nivel para aplicaciones más avanzadas que introducen sus propios tipos de datos. Por ejemplo el uso de editores listos para usar pre empaquetados ofrecen edición de texto plano, edición de texto enriquecido, formularios (elementos de esquema) de edición, y entrada de contraseña.

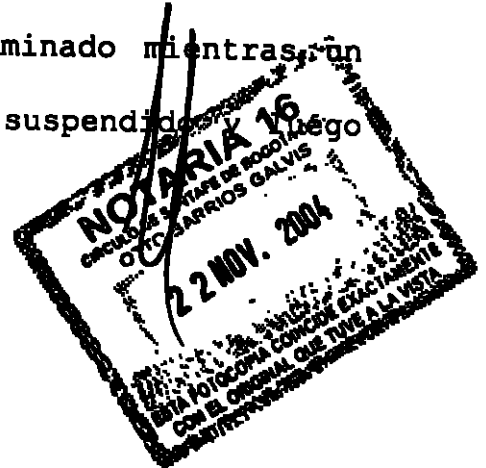
El espacio de nombres de diseño 320 suministra una aproximación flexible y extendible a organizar varias funcionalidades. En lugar de suministrar un editor de características fijas comunes, el espacio de nombres de diseño 320 suministra una combinación de servicios, comportamientos y abstracciones que permiten a los desarrolladores construir soluciones de edición de información especializada y fácilmente personalizable. El



espacio de nombres de diseño 30 incluye varias características de edición personalizables, incluyendo: servicios de alcance, comportamientos de pilas, las pruebas de diseño editores, de selladores abstractos para elementos de edición, modelos de objetos de texto abstracto, capa de adornado, superficie de diseño, protocolo de transferencia de datos generalizado, y mecanismo extensible para deshacer.

Los zervicios de alcance permiten a la asociación de servicios específicos con partes particulares y subporciones de los datos de la aplicación. Este mecanismo también permite dirigir servicios dentro y fuera de diferentes alcances. La clase principal que suministra soporte para servicios es la ServiceManager; servicios individuales son implementados usando las Interfaces IService y IscopedService.

El concepto de "comportamiento de pilas" permite que diferentes comportamientos sean activados en periodos de tiempo apropiados. En una modalidad, la activación de comportamiento basado en tiempo en anidada, como un proceso que puede ser iniciado y terminado mientras, un proceso más general es temporalmente suspendido.



835

apropiadamente restaurado. En lugar de la solución de código duro, la aproximación de comportamientos de pila resuelve este problema permitiendo una integración de procesos que no tienen un conocimiento pre definido acerca de los demás. La clase principal que suministra soporte para este mecanismo es `EditRouter`; Comportamientos individuales son subclases de `Editor` y de `EditBehavior`. Los tipos contruidos de datos editables son soportados por la correspondiente subclase de `EditBehavior` tales como `TextEditor`, `ElementEditor` y `MoveBehavior`.

El concepto "editor diseñador de patrón" permite la separación entre funcionalidad de edición genérica y un subtipo más específico de información editable. El editor no espera una estructura de datos específica sobre la cual operar. En cambio, él asume que el contenido de datos está expuesto a través de alguna Interfaz abstracta, ocultando los detalles de implementación, pro suficiente para la acción correspondiente de editar interacción. Este patrón es particularmente útil con elementos (en formas de edición) y para texto (edición de texto enriquecido).



"Abstract designer for element editing" es una aplicación de el "editor-designer pattern" para un área de edición de elementos (por ejemplo., edición de formas). Varios elementos pueden tener diferentes estructuras internas, apariencias y comportamientos, pero si ellas tienen algún estándar para el movimiento "diseñadores", rotación, o cambio de tamaño, entonces ellas serán evitables por el editor de elementos.

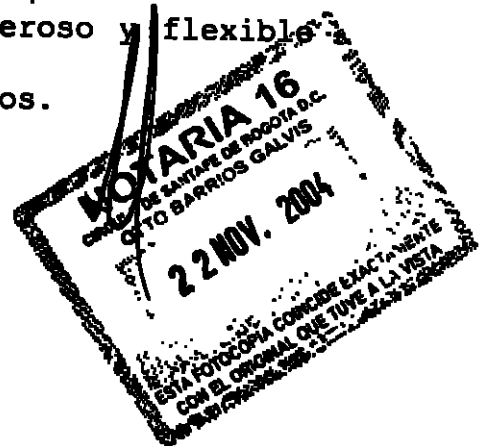
El modelo de texto abstracto es otra aplicación de "editor-designer pattern" para usar en edición de texto enriquecido. Esta abstracción permite diferente apoyo de almacenamiento para participar en edición de texto enriquecido, tal como RTF, XHTML, texto plano, XAML, código sintácticamente resaltado, etc. Aunque la naturaleza semántica la estructura interna de estos tipos pueden ser diferente, exponiéndose a si mismo por un modelo de texto abstracto que le permite la aplicación de mecanismos de edición genéricos. Adicionalmente, esta aproximación permite la integración de datos y texto, tal como el texto de diferente apoyo de almacenamiento puede contener piezas de otro, mientras suministra la misma experiencia de edición.



837

La capa de adorno es una herramienta unificada que suministra una retroalimentación visual enriquecida durante la edición, La capa de adorno suministra soporte uniforme para muchos tipos de retroalimentación. El poder visual puede ser dinámicamente adjuntado o desadjuntado de los elementos y otras porciones de datos para indicar actividad del usuario potencial aplicable a ella. La superficie de diseño es un control XAML que reempaqueta recursos de edición, tales como la capa de adorno y el enrutador de edición. Esto simplifica el desarrollo de edición a nivel intermedio.

Las funciones de edición tales como cortar/copiar/pegar y desplazar/soltar son importantes en un ambiente de información integrado, pero pueden ser difíciles de soportar debido a la complejidad e incompatibilidad de diferentes tipos. El protocolo generalizado de transferencia de datos direcciona esta dificultad suministrando para la combinación de abstracciones usadas para la extracción de datos, inserción de datos, arrastre de datos, y conversión de datos. Este protocolo de transferencia de datos generalizado suministra un mecanismo poderoso y flexible para la interacción de aplicaciones y datos.



El mecanismo extensible deshacer maneja una pila de elementos a deshacer coleccionando datos de diferentes tipos. El administrador de deshacer incluye protocolos para manejar las operaciones deshacer con diferentes tipos.

Las siguientes tablas listan miembros de ejemplos puestos por el espacio de nombre System.Windows.Design namespace.

Clases

ActivationEventFilter

EventFilter

ActivationRectangleAdornerInfo

Esta Es una realimentación que muestra el borde sobre los elementos de DesignActive elements

AdornerInfo

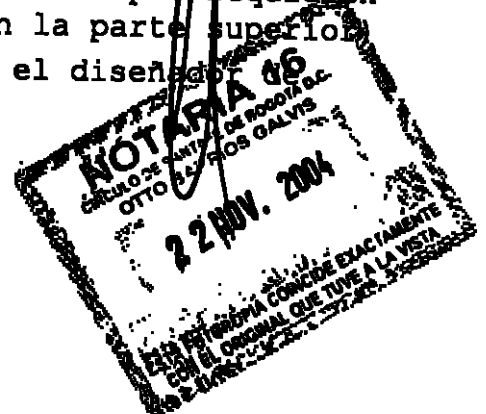
Suministra información acerca de un adorno específico en un adornerset.

AdornerLayer

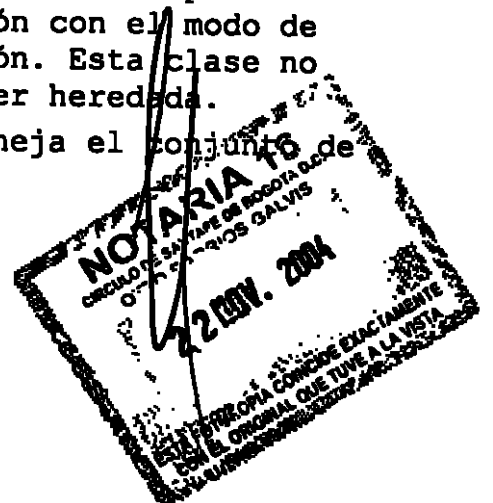
Suministra una superficie para mostrar las decoraciones sobre elementos que requieren trascender el orden Z (los elementos siempre requieren estar en la parte superior)

CanvasDesigner

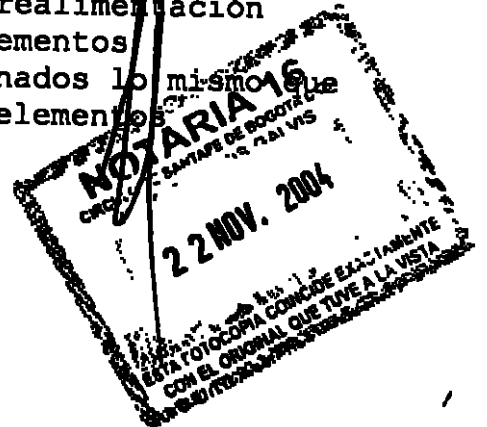
Es el diseñador



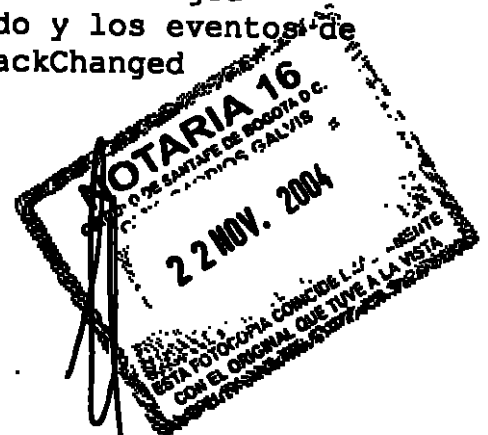
	cuadro por defecto.
ComponentChangeService	Suministra la funcionalidad deshacer y rehacer para las acciones en tiempo de diseño que afecten los componentes.
Designer	Suministra un medio de editar contenido en vivo.
DesignerFilterService	Define un servicio opcional que las aplicaciones y los controles pueden implementar para sobreponerse al mapeado por defecto entre un elemento y el diseñador siempre que suministrado por el DesignerLookupService.
DesignerLookupService	Permite editar comportamientos para mapear elementos para los diseñadores.
DesignSurface	El elemento de raíz recomendado para el contenido editable. Este agrega adornos, servicios y eventos.
EditBehavior	Define la implementación para todos los comportamientos editados.
Editor	Maneja la entrada de usuario para un tipo de selección particular, tal como texto, elemento o tabla. Típicamente existe un editor específico para cada tipo de selección. type.
EditorTypeAttribute	Asocia un tipo de selección con el modo de selección. Esta clase no puede ser heredada.
EditRouter	Maneja el conjunto



	EditBehaviors.activo actualmente.
EditRouterChangedEventArgs	Argumentos para el evento cambiado.
ElementEditor	El Editor de elemento.
EventFilter	Permite el filtrado de los eventos y la asignación.
FrameworkElementDesigner	El diseñador por defecto para todos los controles que se derivan del FrameworkElement.
MoveBehavior	Mover el comportamiento.
ObjectSelection	Selección del sitio.
ParentUndoUnit	ParentUndoUnit Implementación del ITextDesigner para el TextPanel y otros controles que soportan el TextDocumentResult.
RichTextDesigner	RoutedEventAdapter pensó un eventHandler específico posterior a un RoutedEventHandler de modo que un manejador o handler puede manejar diferentes tipos de eventos. Usado por el EditRouter
RoutedEventAdapter	SelectionRouter controla la selección mezclada con múltiples modos de selección. Este deriva del EditRouter para enrutar eventos para los editores modulares (que se derivan del editor).
SelectionRouter	Suministra acceso programático a la selección de elementos UI. Se puede obtener realimentación sobre elementos seleccionados lo mismo que cambiar elementos.
SelectionService	

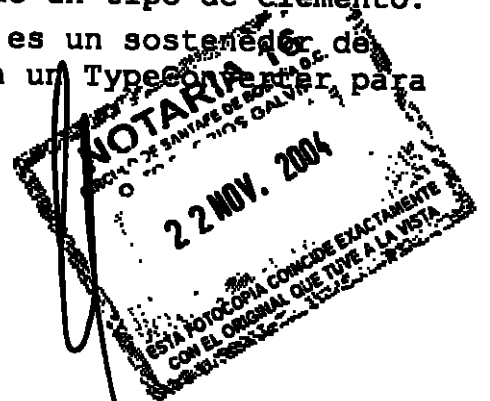


	seleccionados y sus propiedades.
ServiceConverter	ServiceConverter - clase de convertidor para convertir entre una cadena y una instancia de IService.
SimpleTextDesigner	La implementación de ITextDesigner para texto y otros controles que soportan el ITextParagraphResult.
TextDecorator	TextDecorator define un objeto que maneja la decoración de texto a partir de servicios de competencia. Usando TextDecorator en lugar de scriir los valores de propiedades privadamente asegura que la decoración competitiva nunca se sobrepondrá aún en el caso de prioridad idéntica.
TextEditor	Suministra los servicios de selección de texto.
TextSelection	La clase TextSelection encapsula el estado de selección para la clase TextEditor Esta no tiene constructor público, pero está expuesta por vía de una propiedad pública sobre el TextEditor.
UndoService	Implementación de estructura de UndoService
UndoStackChangedEventArgs	Suministra datos para el UndoStackChanged cambiado y los eventos de RedoStackChanged



Interfaces.

IAddRemoveChildDesigner	La Interfaz IAddRemoveChildDesigner es usada para agregar/remover hijos. Suministra acceso a métodos que le permiten a usted recuperar los objetos necesarios para dibujar adornos sobre los FrameworkElements.
IAdornerDesigner	El elemento interior en DesignActivation que puede volverse designActive. Tiene el nombre contenedor ya que se asume que si un elemento es DesignActive, entonces lo más probable es que estará editando algún contenido hijo.
IDesignActivationContainer	Un IDesignActivationContainerParent puede ser huésped de los contenedores y IDesignActivationContainer's in en un ambiente de diseño.
IDesignActivationContainerParent	Interfaz para ser implementada por cualquier elemento que desea participar en la edición.
IDesignSurface	Interfaz de mover el diseñador.
IMoveDesigner	Interfaz IParentUndoUnit
IParentUndoUnit	IScopedService
IScopedService	Interfaz de base para modos de selección.
ISelection	Interfaz de diseñador asociada con el SelectionBehavior (SelectionRouter). Una clase que implementa esto estaría especificada en el atributo Designer de un tipo de elemento.
ISelectionDesigner	Este es un sostenedor de lugar para un TypeContainer para
IService	



	servicios de edición.
	Unos componentes de.
ITextDesigner	Interfaz que se implementan para suministrar el servicio TextEditor sin formación específica de caas y acciones.
IUndoService	Interfaz IUndoService
IUndoUnit	Interfaz IUndoUnit

Enumeraciones

AdornerInfoZOrder	Indicadores de orden Z.
AdornerRenderTriggers	Los indicadores indican las condiciones que causa que los adornos o los ordenadores se vuelvan a reproducir.
EditRouterChange	Enumeración de posibles acciones de cambio de enrutador.
TextDecorationPriority	El TextDecorationPriority clasifica la importancia relativa de diversos marcadores de texto. Es usado para correr la decoración de orden Z en la clase TextDecorator.
UndoState	Enumeración para el estado del manejador deshacer

Delegados

UndoStackChangedEventHandler	Representa el método que manejará un evento UndoStackChanged o RedoStackChanged



844

Espacio de nombre de entrada

Un espacio de nombre de entrada 322 incluye un manejador de entrada que coordina las entradas recibidas por el sistema. El nombre de espacio de entrada 322 también incluye clases que ayudan a manejar y suministran el control para diferentes dispositivos de entradas tales como un teclado o un ratón. El sistema de entrada permite que las aplicaciones obtengan información desde los dispositivos de entrada tal como un ratón, teclados, punzones (lápices). La mayor parte de la funcionalidad de entrada está en la forma de propiedades, métodos y eventos sobre los `UIElement`, `FrameworkElement`, `ContentElement`, y `ContentFrameworkElement` como clases en el nombre de espacio del sistema Windows. El `UIElement` y el `ContentElement` tienen funcionalidad de entrada similar, la cual es capturada por la Interfaz `IInputElement`. Similarmente, el `FrameworkElement` y `ContentFrameworkElement`, los cuales se derivan del `UIElement` y `ContentElement` respectivamente, comparten muchas entradas API y ambos implementan el `IFrameworkInputElement`.



El sistema de entrada suministra soporte completo para los teclados, ratones y punzones. La funcionalidad de teclado incluye eventos de clave abajo/arriba o tecla abajo tecla arriba, manejo del foco y notificación, estado de tecla actual de acceso, y opresión de teclas y convertir las opresiones de teclas en cadena de entrada. La funcionalidad de ratón incluye la posición de ratón, oprimir el botón, el movimiento, eventos de entrada/salida y límites de elementos, captura del ratón, revoloteado y rueda del ratón. La funcionalidad de punzón incluye posición y movimiento (ambos cuando el punzón está tocando la superficie y cuando está "al aire"), tocar/clicks, captura, reconocimiento de gestos.

El espacio de nombre de entrada contiene clases de ayudador auxiliares necesarios para la funcionalidad anterior, tal como las enumeraciones, argumentos de eventos, firmas de delegados, etc. Adicionalmente, el espacio de nombre de entrada incluye las clases de teclado, ratón, y punzón, lo que suministra información relacionada con el estado actual de estos dispositivos. El sistema de entrada suministra maneras de especificar los cursores de ratón sobre un elemento particular, sobre una aplicación completa.



846

El sistema de entrada está integrado con el Text Services Framework (TSF), el cual permite que otros software de entrada entiendan el contexto textual en el cual el nuevo texto está siendo ingresado. Esto es usado por editores de método de entrada (IME), los cuales permiten a los usuarios de lenguajes de Asia del Este convertir los diferentes golpes de tecla en un carácter único, el El software IMS comunica con el TSF para identificar la mejor interpretación de los golpes de tecla con base en el contexto y usa TSF para insertar el texto en el documento.

Similarmente, el reconocimiento de voz utiliza STF para escoger el mejor reconocimiento que insertarlo dentro del documento.

El sistema de entrada ofrece integración automática con el TSF, en la forma de un evento TextInput. El espacio de nombre de entrada también ofrece a los API para describir el contexto textual, y controlar la conversión de entrada de teclado basto en entrada textual. La clase InputManager suministra filtros monitores lo que permite a una tercera parte observar



corriente de entrada y crear nuevos eventos de entrada antes de los eventos de entrada originales que han sido disparados. El `inputManager` también suministra los API para nuevos dispositivos que sean enchufados en el sistema de entrada, usando el `IInputProvider` y el `InputReport`.

La siguiente lista de tablas de miembros ejemplos expuestos por el `System.Windows.Input` namespace.

Clases

`AccessKeyManager`

`AccessKeyManager` expone a funcionalidad de acceso a la tecla, también conocido como nemotécnicos. Un ejemplo típico es el botón OK con el alt-O siendo el nemotécnico - presionar alt-O es lo mismo que hacer clic del botón.

`Cursor`

Representa el cursor de ratón para uso.

`CursorTypeConverter`

Convierte los cursores de ratón a y desde cadenas.

`FocusChangedEventArgs`

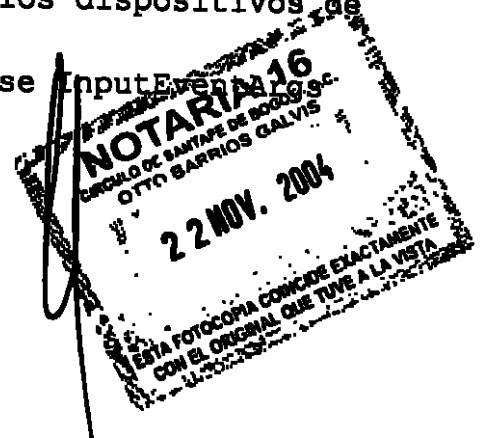
La clase `FocusChangedEventArgs` contiene información acerca de los cambios de foco.

`InputDevice`

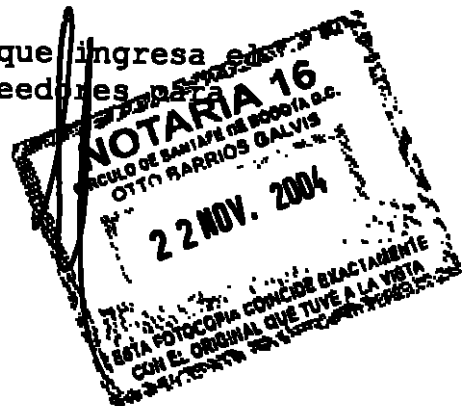
Suministra la clase de base para todos los dispositivos de entrada.

`InputEventArgs`

La clase `InputEventArgs`



	representa un tipo de <code>RoutedEventArgs</code> que son relevantes para todos los eventos de entrada.
<code>InputLanguageChangedEventArgs</code>	La clase <code>InputLanguageEventArgs</code> representa un tipo de <code>RoutedEventArgs</code> que son relevantes para eventos resurgidos para indicar un cambio en el lenguaje de entrada (humano).
<code>InputLanguageChangingEventArgs</code>	La clase <code>InputLanguageEventArgs</code> representa un tipo de <code>RoutedEventArgs</code> que son relevantes a eventos surgidos para indicar un cambio en el lenguaje de entrada (humano).
<code>InputLanguageEventArgs</code>	La clase <code>InputLanguageEventArgs</code> representa un tipo de <code>RoutedEventArgs</code> que son relevantes a eventos surgidos para indicar un cambio en el lenguaje de entrada (humano).
<code>InputLanguageManager</code>	La clase <code>InputLanguageManager</code> es responsable de manejar el lenguaje de entrada en Avalon.
<code>InputManager</code>	La clase de <code>InputManager</code> es responsable de coordinar todo lo del sistema de entrada en in "Avalon".
<code>InputMethod</code>	La clase <code>InputMethod</code> expone los API relacionados con TSF, que están comunicandose o accedendo las propiedades TIP.
<code>InputMethodStateChangedEventArgs</code>	Esta clase <code>InputMethodStateChangedEventArgs</code> es usada cuando el editor de método de entrada (IME) cambia su estado.
<code>InputProviderSite</code>	El objeto que ingresa al uso de los proveedores de entrada.



InputReport

reportar la entrada al manejador de entrada.

El InputReport es una clase de base abstracta para todas las entradas que son reportadas al InputManager.

InputReportEventArgs

Suministra datos para el evento que ha surgido cuando un InputReport está siendo procesado.

Keyboard

La clase Keyboard expone los API relacionados con el teclado.

KeyboardDevice

La clase KeyboardDevice representa el dispositivo de teclado a los miembros de un contexto.

KeyboardEventArgs

La clase KeyboardEventArgs suministra acceso a dispositivo apuntador lógico para todos los argumentos de eventos derivados.

KeyEventArgs

La clase KeyEventArgs contiene información acerca de los estados de la tecla.

KeyInterop

Suministra métodos estáticos para convertir entre Win32 VirtualKeys y la enumeración de tecla "Avalon".

Mouse

La clase Mouse expone los API relacionados con el ratón.

MouseButtonEventArgs

El MouseButtonEventArgs describe el estado de un botón de ratón.

MouseDevice

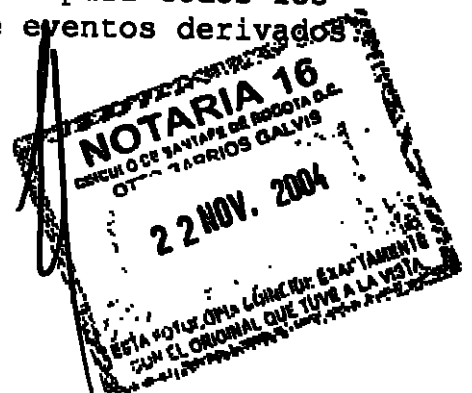
La clase MouseDevice representa al dispositivo de ratón a los miembros de un contexto.

MouseDoubleClickEventArgs

Suministra datos para los eventos que son surgidos cuando se hace doble clip en el ratón.

MouseEventArgs

La clase MouseEventArgs suministra acceso al dispositivo lógico de ratón para todos los argumentos de eventos derivados.



830

MouseWheelEventArgs

El **MouseWheelEventArgs** describe el estado de una rueda de ratón.

NotifyInputEventArgs

Suministra información acerca del evento de entrada que está siendo procesado por el manejador o administrador de entrada.

PreProcessInputEventArgs

Permite al manejador cancelar el proceso de un evento de entrada.

ProcessInputEventArgs

Suministra acceso al área de etapas del administrador de entrada.

RawKeyboardInputReport

La clase **RawKeyboardInputReport** encapsula la entrada directa provista desde un teclado.

RawMouseInputReport

La clase **RawMouseInputReport** encapsula la entrada directa provista desde un ratón.

StagingAreaInputItem

Esta clase encapsula un evento de entrada mientras está siendo procesado por el administrador de entrada.

TextInputEventArgs

La clase **TextInputEventArgs** contiene una representación de texto de entrada.

TextManager

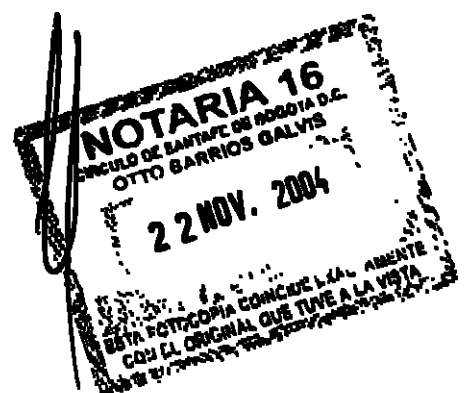
La clase **TextManager** suministra la promoción de evento entrada al texto.

TextServicesContext

Esta clase maneja la interoperación entre un **UIDispatcher** y la estructura de servicios del texto, un **Services COM API** nativo que permite la entrada **IME Asia del Este**.

TextStoreInfo

Esta es una clase protegida de enlace de solicitud que es interna.



Interfaces

IInputLanguageSource	Una Interfaz para controlar el lenguaje de fuente de entrada.
IInputProvider	Una Interfaz implementada para los proveedores de entrada.
IKeyboardInputProvider	Una Interfaz para controlar las entradas por teclado.
IMouseInputProvider	Una Interfaz para controlar entrada proveida por el ratón.

Enumeraciones

CursorType	Una enumeración de los tipos de cursor soportados.
ImeConversionMode	ImeConversionMode
ImeSentenceMode	ImeSentenceMode
InputMethodState	Estado del Ime
InputMode	El modo de procesamiento de entrada cuando fue suministrada la entrada.
InputType	El tipo de entrada que está siendo reportado.
Key	Una enumeración de todos los posibles valores de tecla en un teclado.
KeyState	La enumeración KeyState describe el estado en que las teclas del teclado pueden estar.
ModifierKeys	La enumeración ModifierKeys describe un conjunto de teclas comunes usadas para modificar otras operaciones de entrada.



852

852

La enumeración MouseButton describe los botones disponibles en el dispositivo de ratón.

La enumeración MouseButtonState describe los posibles estados de los botones disponibles en el dispositivo de entrada de ratón.

Las acciones directas que están siendo reportadas desde el teclado.

Las acciones directas que están siendo reportadas desde el ratón.

Modo de voz.

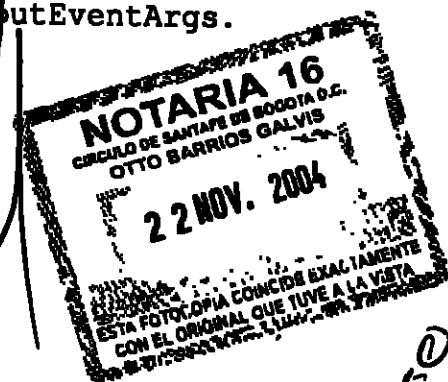
Estructuras

Versión administrada de la estructura Win32 MSG.

Delegados

El delegado para usar por los manejadores que reciben FocusChangedEventArgs.

El delegado para usar por los manejadores que reciben InputEventArgs.



250

InputLanguageEventHandler	Este es un delegado para los eventos InputLanguageChanged y InputLanguageChanging.
InputMethodStateChangedEventHandler	El delegado para usar por los manejadores que reciben el estado del método de entrada del evento cambiado.
InputReportEventHandler	El delegado para usar por los manejadores que reciben PointerMoveEventArgs.
KeyboardEventHandler	El delegado para usar por los manejadores que reciben KeyboardEventArgs.
KeyEventHandler	El delegado para usar por los manejadores que reciben KeyEventArgs.
MouseButtonEventHandler	El delegado para usar por los manejadores que reciben MouseButtonEventArgs.
MouseDoubleClickEventHandler	El delegado para usar por los manejadores que reciben MouseDoubleClickEventArgs.
MouseEventHandler	El delegado para usar por los manejadores que reciben MouseEventArgs.
MouseWheelEventHandler	El delegado para usar por los manejadores que reciben MouseWheelEventArgs.
NotifyInputEventHandler	Tipo de delegado para manejar eventos que usan NotifyInputEventArgs.
PreProcessInputEventHandler	Tipo de delegado que maneja eventos que usan PreProcessInputEventArgs.
ProcessInputEventHandler	Tipo de delegado que manejan eventos que usan ProcessInputEventArgs.



El delegado para usar
por manejadores que
reciben
TextInputEventArgs.

TextInputEventHandler

Espacios de Nombres de Navegación

Un espacio de nombre de navegación 324 suministra un conjunto de clases y servicios que le permiten construir aplicaciones con paradigmas de navegación, tal como una aplicación de buscador. Estas clases y servicios permiten el desarrollo de aplicaciones con experiencias de navegación adaptadas. Por ejemplo, cuando se compra un producto o un servicio desde una tienda en línea, hacer click en un botón "Back" causa que la aplicación muestre una página diferente que le pregunta al usuario si quiere cancelar o cambiar su orden. En otro ejemplo, activar un botón "Refresh" causa que una aplicación recupere nuevos datos en lugar de recargar primeramente la aplicación seguida por la recuperación de los nuevos datos. El espacio del nombre de navegación 324 también incluye funciones de página que suministran un mecanismo para generar una jerarquía de preguntas que son presentadas a un usuario.



855

Los siguientes miembros ejemplo de la lista de tablas expuestas por System.Windows.Navigation namespace.

Clases

BoolWrapper

Una carpeta para un valor Boleano.

BoolWrapperPageFunction

Un tipo de función de página que regresa un valor Boleano a la página anterior.

BoolWrapperReturnEventArgs

IntWrapper

Una carpeta para el valor Int 32.

IntWrapperPageFunction

Una función de página escrita que regresa un valor Int32 a la página anterior.

IntWrapperReturnEventArgs

Journal

Contiene una historia de navegación de la aplicación.ory.

JournalEntry

Representa una entrada Journal..

LoaderService

Usada para establecer el cargador actual en un abdomen dado.

NavigateEventArgs

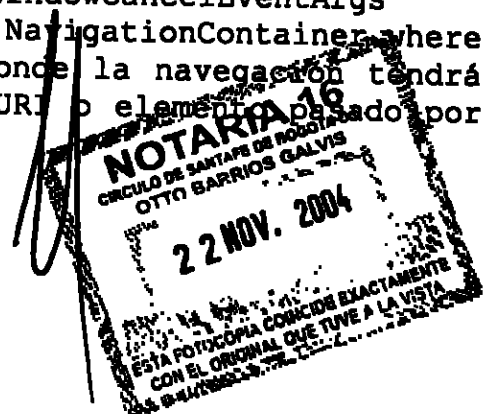
Obsoleto.

NavigatingCancelEventArgs

Argumentos de evento para la NavigationApplication.Navigating and NavigationWindow.Navigating events.

NavigatingNewWindowCancelEventArgs

Argumentos de evento para el evento cancelable NavigatingNewWindow el ent e NavigatingNewWindowCancelEventArgs especifica el NavigationContainer where objetivo en donde la navegación tendrá lugar con el URI o elemento pasado por



defecto la propiedad de cancelación es establecida en falso. El establecimiento de verdadero que la nueva ventana sea abierta, y la navegación no tendrá lugar..

NavigationApplication

Representa una aplicación de navegación tipo "Cuernilarga" al sistema.

NavigationContainer

Una región navegable que puede contener un árbol de marcación "Avalon". Normalmente, esta clase no es usada directamente pero puede ser usada como clase variante para una implementación de adaptación.

NavigationErrorCancelEventArgs

Contiene los argumentos para los eventos NavigationApplication.NavigationError y NavigationWindow.NavigationError

NavigationEventArgs

Argumentos de evento para eventos de navegación no cancelables, que incluyen LoadCompleted, LoadStarted, Navigated, and NavigationStopped.

NavigationProgressEventArgs

Contiene los argumentos para los eventos NavigationApplication.NavigationProcess and NavigationWindow.NavigationProcess

NavigationService

Contiene los delegados usados por los eventos de navegación, y una propiedad dinámica que contiene una interfaz INavigator.

NavigationWindow

Representa una ventana de navegación.

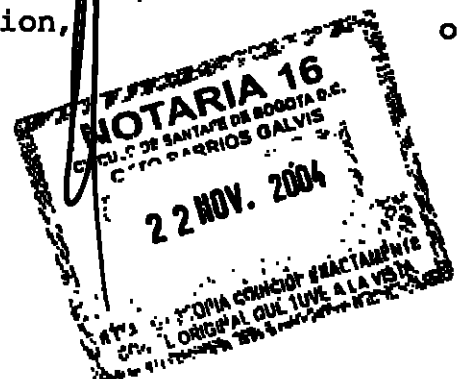
ObjectPageFunction

Una función de página escrita que regresa un valor objeto a la página previa.

ObjectReturnEventArgs

PageFunction

Esta clase no está soportada directamente. En su lugar usa una de las clases escritas: BoolWrapperPageFunction, IntWrapperPageFunction, ObjectPageFunction,



ReturnArgs

StringPageFunction.

El objeto de argumentos de evento para el evento Return. Esta clase no está soportada directamente. En su lugar el uso de los argumentos Return para la clase escrita apropiada : BoolWrapperReturnEventArgs, IntWrapperReturnEventArgs, ObjectReturnEventArgs, StringReturnEventArgs.

StringPageFunction

Una función de página escrita que regresa un valor de cadena a la página previa.

StringReturnEventArgs

WindowNavigationContainer

Representa una region navegable dentro de una ventana de navegación.

Interfaces

IJournalData

La Interfaz IjournalData-debe ser implementada por controles que requieren persistir en el estado en el journal, y recuperar o restaurarlo cuando la página es revisitada.

ILoader

Interfaz usada para resolver los UI en corrientes. Este puede ser usado para cargar el contenido desde el archivo, http, contenedor y administrador y desadministrador de recursos.

INavigator

Implementado por los contenedores de navegación para suministrar acceso a las propiedades, métodos, y eventos que soportan la navegación.

INavigatorService

La Interfaz InavigatorService. Esta Interfaz estará disponible sobre el NavigationWindow que permitiendo



implementador de INavigator se registre el mismo para participar en objetivos Hyperlink hiperenlace.

Enumeraciones

NavigationMode Usada para especificar el modo de navegación.

Delegados

BoolWrapperReturnEventHandler

IntWrapperReturnEventHandler

LoadCompletedEventHandler

LoadStartedEventHandler

NavigatedEventHandler

NavigateEventHandler

NavigatingCancelEventHandler

NavigatingNewWindowCancelEventHandler

NavigationErrorCancelEventHandler

Representa el método que maneja los eventos

NavigationApplication.LoadCompleted y NavigationWindow.LoadCompleted events.

Representa el método que maneja el evento LoadStarted.

Representa el método que maneja los eventos NavigationApplication.Navigated y NavigationWindow.Navigated events.

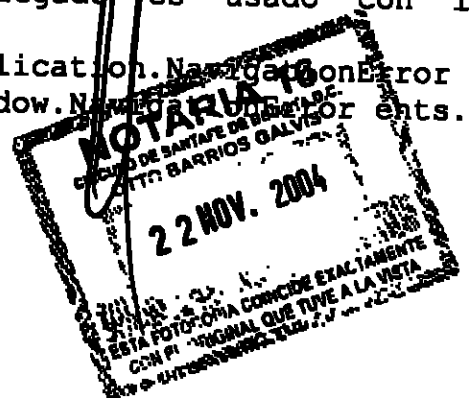
Obsoleto.

Representa el método que maneja los eventos NavigationApplication.Navigating y NavigationWindow.Navigating events.

Representa el método que maneja el evento NavigatingNewWindow.

Este delegado es usado con los eventos

NavigationApplication.NavigationError y NavigationWindow.NavigationError events.



NavigationProgressEventHandler	Este delegado es usado con los eventos NavigationWindow.NavigationProgress and NavigationApplication.NavigationProgress .
NavigationStoppedEventHandler	Representa el método que maneja los eventos NavigationApplication.NavigationStopped y NavigationWindow.NavigationStopped .
ObjectReturnEventHandler	Representa el método que maneja el evento Return. Esta clase no está directamente soportada. En su lugar se usa el manejador de evento return para la clase escrita apropiada: BoolWrapperReturnEventHandler, IntWrapperReturnEventHandler, ObjectReturnEventHandler, or StringReturnEventHandler.
StringReturnEventHandler	

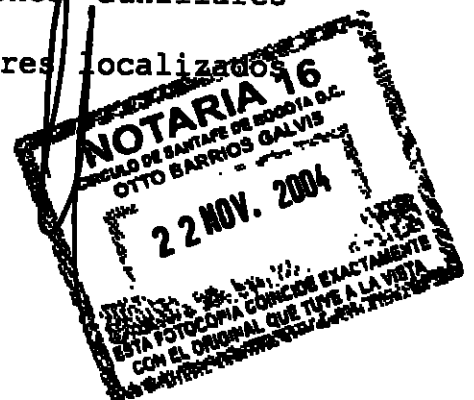
Espacios de nombre de Automatización

Un espacio de nombre de automatización 326 contiene miembros usados para soportar la accesabilidad y la automatización de la Interfaz de Usuario. El sistema de accesabilidad incluye un lado de cliente y un lado de proveedor. Un sistema de herramientas incluye herramientas de automatización del lado del cliente que incluyen una clase de automatización de cliente para buscar información de Interfaz de usuario. La clase de automatización de cliente incluye herramientas



registro de evento y herramientas de descubrimiento de elemento lógico. El conjunto de herramientas adicionalmente incluye las herramientas de automatización del lado de proveedor para suministrar al cliente con información de Interfaz de usuario. Las herramientas de automatización de lado de proveedor incluyen una clase de automatización de proveedor que tiene herramientas para suministrar al cliente con información de evento.

Una clase de cliente de automatización suministra métodos de automatización UI para uno o más clientes. La clase de automatización de cliente contiene métodos que no son específicos para ningún elemento UI. La clase de automatización de cliente puede suministrar un método para obtener un elemento lógico o directo desde un punto, un manejador de ventana, o un elemento de raíz de escritorio. La clase de automatización de cliente adicionalmente puede suministrar métodos para encontrar un elemento lógico con base en criterios de entrada. La clase de automatización de cliente preferiblemente también incluye un método para registrar y desregistrar notificaciones de eventos. La clase de automatización preferiblemente también suministra funciones auxiliares para cargar DLL, proxys recuperar los nombres localizados



de propiedades y patrones de control y llevar a cabo comparaciones de elementos. La clase de automatización de cliente también incluye métodos para que los clientes escuchen eventos. Los miembros de ejemplo en la lista de las tablas siguientes se exponen por el System.Windows.Automation namespace.

Clases

ApplicationWindowPattern

Expone el comportamiento y la información típicamente asociada con una ventana de aplicación de nivel superior. Los clientes pueden usar esta clase para apilar o para mostrar en filas o en cascada los hijos de Interfaz de documentos múltiples de aplicación (MDI), encuentra su botón sobre las barras de tareas, y localiza secciones bien conocidas de su Interfaz de usuario tales como barras de herramientas y menús.

AssociatedInformationPattern

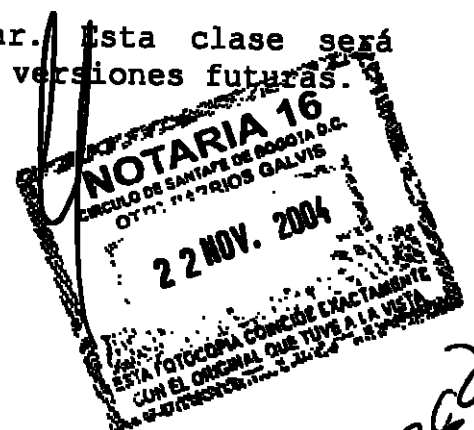
Expone la semántica y metadatos para los elementos UI que representan otros objetos..

Automation

Contiene los métodos de automatización de la Interfaz de Usuario (UI) para clientes (tecnología de ayuda o manuscrito de ensayo automatizado). Estos métodos no son específicos a un elemento particular.

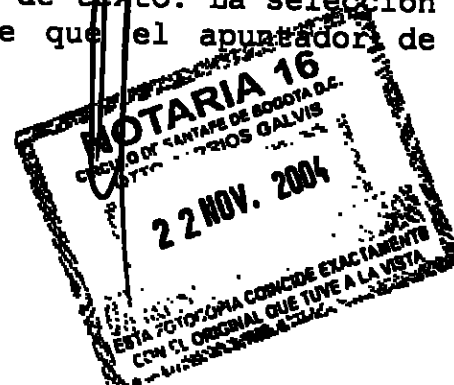
AutomationEvent

No usar. Esta clase será removida en versiones futuras.



260

AutomationEventArgs	Clase de argumentos de patrón o de eventos adaptados.
AutomationFocusChangedEventArgs	Clase de argumentos de foco de evento.
AutomationIdentifier	Clase base para identificadores con base en identidad de objeto. Esta clase es efectivamente abstracta; solamente clases derivadas son mencionadas.
AutomationIdentifierProxy	Clase interna usada para deserializar AutomationIdentifiers. Estos no deben ser usados directamente.
AutomationPattern	No usar. Esta clase será removida en versiones futuras.
AutomationPermission	Suministra un conjunto de permisos para acceder elementos UI. Esta clase no puede ser heredada.
AutomationPermissionAttribute	Suministra métodos y propiedades para el AutomationPermissionAttribute.
AutomationProperty	No usar. Esta clase será removida en futuras versiones.
AutomationPropertyChangedEventArgs	Clase de argumentos de eventos de PropertyChanged.
AutomationTextAttribute	Identificador para automatización de atributos de texto.
AutomationTextPointer	Representa la posición de un caracter dentro del texto. El AutomationTextPointer suministra métodos y propiedades para acceder texto y navegar en texto.
AutomationTextRange	Usado para obtener, establecer, agregar y remover la selección.
AutomationTextSelection	Propósito: El objeto AutomationTextSelection maneja todas la administración de selección de texto. La selección en la que que el apuntador de



863

AutomationTextUnit

inserción está o está la selección activa. Ejemplos de uso. Es usado cuando los clientes quieren agregar, remover, modificar o establecer una selección. Los clientes pueden también encontrar lo que está seleccionado actualmente por medio del AutomationTextSelection.

Identificador para la automatización de unidades de texto.

BasePattern

Clase interna.

DockPattern

Expone una habilidad del elemento para cambiar su estado de archivo en tiempo de corrida.

ElementNotEnabledException

Esta excepción es lanzada cuando el código de cliente intenta manipular un elemento o control que actualmente no está habilitado.

ElementPath

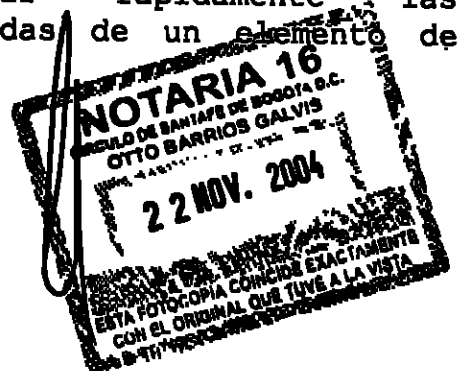
El ElementPath suministra criterios necesarios para regresar subsecuentemente al elemento lógico anteriormente registrado, modificado, o creado completamente por el proveedor de la aplicación.

ExpandCollapsePattern

Expone una habilidad de control para expandir para mostrar más contenido o para colapsar o para colapsar y el contenido Escondido. Ejemplos incluyen botones de menu, botón de inicio, elemento de vista de árbol en Windows Explorer, y cajas combo.

GridItemPattern

Expone los elementos de una rejilla. Permite a los clientes determinar rápidamente las coordenadas de un elemento de rejilla.



864

GridPattern

Expone la funcionalidad básica de rejilla, tamaño, y navegación a celdas especificadas.

HierarchyItemPattern

Expone la relación de jerarquía entre elementos de la Interfaz de control de Usuario independientes de su relación en el árbol lógico.

Input

Suministra métodos para enviar entradas de ratón y de teclado.

InvokePattern.

Implementada por objetos (controles) que llevan a cabo una acción única y no ambigua. La mayoría de estos controles no mantienen estado; invocarlos inicia una acción en la aplicación. Ejemplos de controles que implementa esta Interfaz incluye los botones cruzadores, los botones de barras de herramienta, los elementos de menú, los hiperenlaces, las cajas de verificación, los botones de radio, los símbolos más en los controles de vista de árbol, y los elementos de vista de lista en Windows Explorer.

LogicalElement

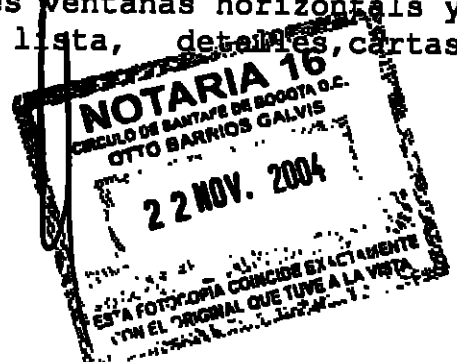
Representa un elemento UI en el árbol lógico sin importar de la implementación ("Avalon", Microsoft® Win32®).

LogicalStructureChangedEventArgs

Clase de argumentos de evento cambiado por estructura lógica.

MultipleViewPattern

Expone una habilidad de los elementos para conmutar entre representaciones multiples del mismo conjunto de información datos o hijos. Los ejemplos incluyen ListView (Miniaturas, multiples ventanas horizontales y conos, lista, detalles, cartas



865

NoClickablePointException

Excel (tipo porque, línea, barras, valor de celda con una formula), y calendario Outlook (Año, Mes, Semana, Día).

La excepción que es lanzada cuando el valor de un argumento está por fuera del rango permitido de valores, según se definió por el GetClickablePoint. Por ejemplo, cuando el rectángulo de límites está vacío, no tiene ancho o alto o el LogicalElement en ese punto no es el mismo que fue llamado.

ProxyAssemblyNotLoadedException

Esta excepción es lanzada cuando existe un problema al cargar un conjunto proxy. Esto puede ocurrir en respuesta a la Automation.RegisterProxyAssembly o cuando se cargan los proxies por defecto cuando el primer LogicalElement de base HWNE se encuentra.

RangeValuePattern

Expone una habilidad del control para manejar un valor dentro de un rango finito. Este lleva a valores máximos y mínimos válidos de control y valor actual.

RawElement

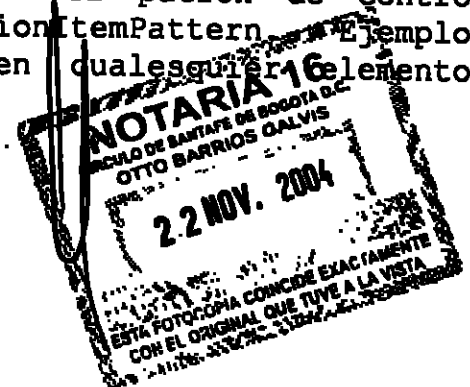
Representa un elemento en el árbol de elementos de partida.

ScrollPattern

Representa elementos UI que expresan un valor.

SelectedItemPattern

Representa elementos individuales en contenedores que manejan la selección. Los clientes de automatización UI usan esta clase para obtener información acerca de, o para manipular elementos UI que soportan el patrón de control SelectionItemPattern. Ejemplos incluyen cualesquiera elementos



SelectionPattern

que puedan ser seleccionados en un control que soporte el patrón de control SelectionPattern, tales como el elemento en una caja de lista o en una vista de árbol.

Representa contenedores que manejan selección. Usado por los clientes de la automatización UI para obtener información acerca de o para manipular, los elementos de Interfaz de usuario que soportan el patrón de control SelectionPattern.

SortPattern

Expone un orden de clasificación actual del contenedor y permiten a los clientes reclasificar elementos programáticamente.

SplitPattern

Representa ventanas que pueden clonarse ellas mismas creando una ventana adyacente.

TableItemPattern

Representa elementos de rejilla con información de encabezamiento.

TablePattern

Representa una rejilla o una tabla que tiene información de encabezamiento.

TextPattern

Representa texto, tal como un control de edición.

TopLevelWindowEventArgs

Clase de argumentos de eventos TopLevelWindowEventArgs

TreeLoadEventArgs

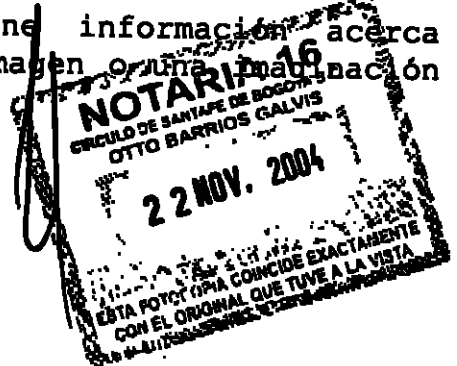
Clase de argumentos de evento TreeLoadEventArgs.

ValuePattern

Expone valores para los controles que no cubren un rango tal como botones de radio, botones de dos posiciones, cajas de verificación, cajas de edición, valores de color RGB, y elemento de menú verificables.

VisualInformationPattern

Expone información acerca de una imagen o una imagen



864

que conlleva o que dirige información al Usuario de modo tal que las notificaciones de onda o globos que giran de Internet Explorer que un documento se está descargando o papeles volando Windows Explorer que indica que el copiado está en progreso.

WindowPattern

Expone la habilidad de un elemento para cambiar su posición en pantalla o tamaño. Ejemplos incluyen ventanas de aplicación de alto nivel (Word o Windows Explorer), ventanas Window principales de Outlook (carpetas, mensajes de , e mail, tareas), y ventanas de hijos MDI.

ZoomPattern

Expone el nivel de magnificación actual en un control y permite que los clientes lo cambien programáticamente.

Enumeraciones

Contiene las notificaciones de banderas AutomationPermissionFlag de acceso para un objeto AutomationPermission.

Diferentes maneras en que los hijos pueden ser dispuestos.

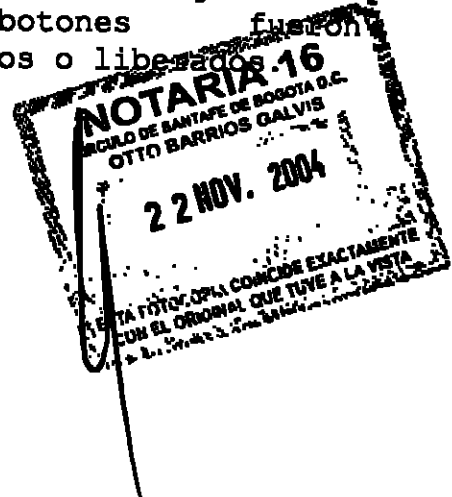
ChildArrangement

El borde del contenedor a la cual la ventana documentable se cuelga.

DockPosition



ExpandCollapseState	Usado para el patrón ExpandCollapse para indicar estados expandidos-colapsados.
GetDataOptions	Opciones para GetData
HierarchyItemState	Enumeración que indica el estado de un elemento de jerarquía: colapsado, expandido, o parcialmente expandido.
ItemCheckState	Enumeración que indica el estado de las cajas de verificación, los botones de radio y controles similares.
LoadState	Banderas de estado de árbol.
LogicalMapping	Valores que indican si un nodo en el árbol de partida estará visualizado en el árbol lógico.
ProxyFlags	Enumeración usada para indicar resultados de solicitar una propiedad.
RowOrColumnMajor	Es el dato en esta tabla que mejor se presenta por fila o columna.
ScopeFlags	Banderas de indicación usadas para definir alcance cuando se escuchan a los eventos.
ScrollAmount	Usado para el ScrollPattern para indicar la dirección (hacia adelante o hacia atrás) y la cantidad (página o línea que se ha pasado).
SendMouseInputFlags	Banderas de indicación para SendMouseInput. Estas banderas indican si el movimiento tuvo lugar o si los botones fueron presionados o liberados.



SplitStyles

Bandera para las direcciones en las cuales una ventana puede dividirse.

StructureChangeType

Banderas que indican cambios en la estructura del árbol lógico.

TextPointerGravity

La gravedad determina la colocación de un AutomationTextPointer cuando el texto es insertado en el AutomationTextPointer. La gravedad es siempre desde el punto de vista de los apuntadores de texto, es decir yo estoy primero que mi carácter o yo estoy después de mi carácter. Cuando el usuario inserta un carácter en su localización de apuntamiento de texto, antes significa que su apuntador estará colocado antes del nuevo carácter, y después significa que el apuntador será colocado después del nuevo carácter.

WindowChangeType

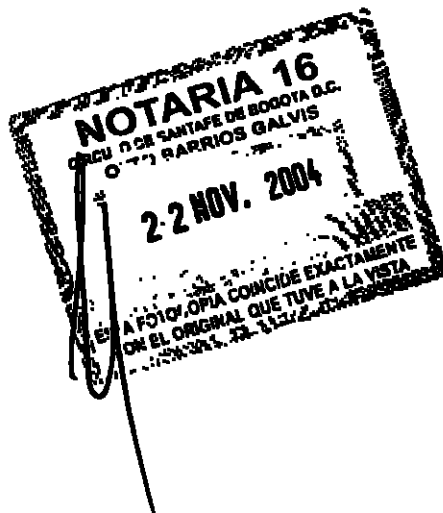
Banderas que indican cambios en las ventanas de nivel superior.

WindowInteractionState

El estado actual de la ventana para la interacción del usuario.

WindowVisualState .

El estado usado para describir el estado visual de la ventana. WindowVisualState sigue a la definición Office y HTML del WindowState.



Estructuras

MatchCondition Especifica los criterios para encontrar elementos usando FindLogicalElement o FindRawElement.

NameAndData Contiene el nombre, el dato como un objeto, y dato como una cadena.

ProxyDescription Estructura que contiene información acerca de un proxy

SortInformation Información usada para clasificar.

Delegados

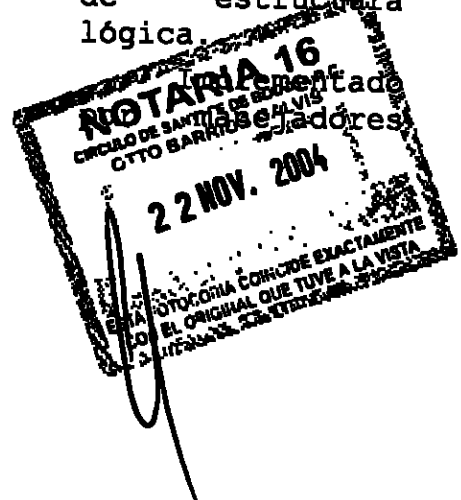
AutomationEventHandler Delegado para manejar AutomationEvents

AutomationFocusChangedEventHandler Delegado para manejar eventos de cambio de foco.

AutomationPropertyChangedEventHandler Delegado para manejar la propiedad de Automation de eventos de cambio.

LogicalStructureChangedEventHandler Delegado para manejar eventos de cambio de estructura lógica.

ProxyFactoryCallback



TopLevelWindowEventHandler

HWND; llamado por la estructura de automatización UI para solicitar un proxy para una ventana especificada. Debe regresar un proxy si está soportado, o anular sino está soportado.

Delegado para manejar eventos de ventana de nivel superior.

Los siguientes miembros de ejemplo en las listas de las tablas se expone por el System.Windows.Automation.InteropProvider namespace.

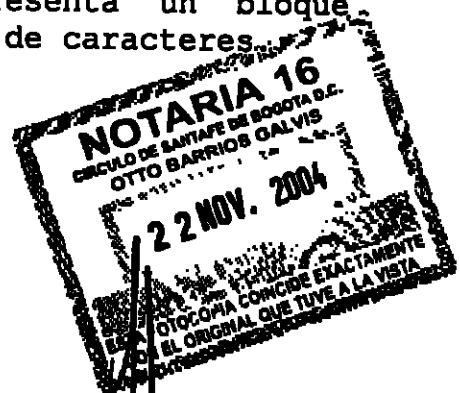
Clases

AutomationInteropProvider

Contiene métodos usados por aplicaciones o controles Win32 que implementan la automatización de Interfaz de usuario (UI). Esta clase no puede ser heredada.

TextPointerPair

Representa un bloque contiguo de caracteres.



Interfaces

IApplicationWindowInteropProvider

Expone el comportamiento y la información típicamente asociada con una ventana de de aplicación del nivel superior.

IAssociatedInformationInteropProvider

Expone la semántica y los metadatos para los elementos UI que representan otros objetos.

IDockInteropProvider

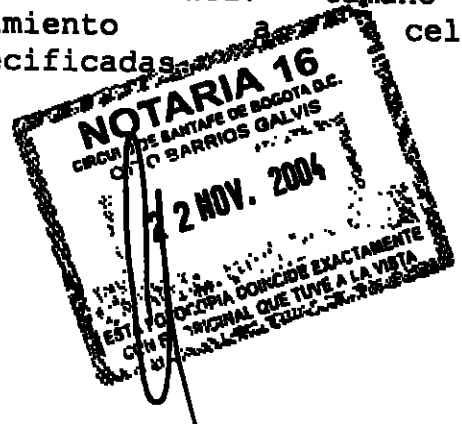
Expone una habilidad de su elemento para cambiar su estado de archivo en tiempo de corrida.

IExpandCollapseInteropProvider

Expone una habilidad del control para expandir para mostrar o visualizar más contenido o colapsar para esconder el contenido. Soportado en conjunto con el patrón HierarchyItem pattern on sobre los elementos TreeView para suministrar un comportamiento similar a árbol, pero es también relevante para controles individuales que abren y cierran. Ejemplos de los UI que implementan esto incluye: elementos TreeView menús inteligentes de Office que han sido colapsados - sed - Chevrons en barras de herramientas-cajas combo-menús "Expandos" en la ventana de tareas de Windows Explorer (lado de mano izquierda donde la vista de la carpeta es frecuentemente desplegada).

IGridInteropProvider

Expone la funcionalidad de rejilla básica: tamaño y movimiento a celdas especificadas.



843

IGridItemInteropProvider

Representa un elemento que está dentro de una rejilla. No tiene métodos, solamente propiedades.

IHierarchyItemInteropProvider

Expone y permite a los clientes atravesar la relación jerárquica entre elementos UI independiente de su relación en el árbol lógico. Las relaciones de jerarquía están por definición no circulares. Ejemplos de UI que implementan esto incluye: - TreeView ítems - Visio Org Chart -menús - controles de Listview controls cuando "muestra en grupos" el modo es activo.

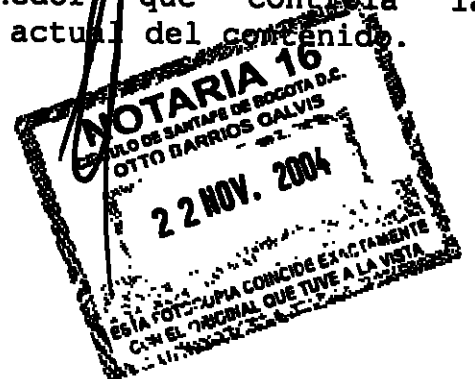
Implementado por objetos (controles) que llevan a cabo una acción única e inambigua. La mayoría de estos controles no mantiene el estado; invocarlos inicia una acción en la aplicación.

IInvokeInteropProvider

Ejemplos de usuarios de elementos de Interfaz que implementan esta Interfaz incluyen los botones de empujar, botones de barra de herramientas, elementos de menú, hiperenlaces, cajas de verificación, botones de radio, el símbolo más en el control de vista de árbol, y los elementos en la vista de lista en el Windows Explorer.

IMultipleViewInteropProvider

Expone una habilidad del elemento para cambiar entre representaciones múltiples del mismo conjunto de información, datos, o hijos. Este patrón debe ser implementado en el contenedor que controla la vista actual del contenido.



844

IRangeValueInteropProvider

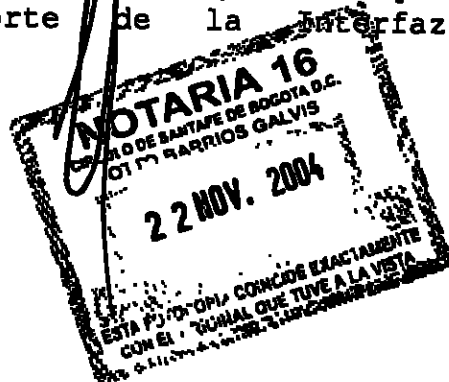
Expone un conjunto relacionado de propiedades que reflejan una habilidad de controles para manejar un valor dentro de un rango finito. Esto lleva a controlar valores validos mínimo y máximo y su valor actual. . Ejemplos: barra de progreso de los Spinners numéricos, control IP (sobre el octeto individual) algunas barras de barrido de colores Pi Pickers algunas Interfaces públicas de diapositivas que representan elementos UI que son expresadas en un valor actual y un rango de valor. En la Interfaz pública tiene la misma definición que el tión as IValueProvider. Las dos diferencias de patrón es que el RangeValue tiene propiedades adicionales y propiedades generalmente que no parecen en las Interfaces de patrón público.

IRawElementProviderFragment

Implementado por los proveedores para exponer elementos que son parte de una estructura que tiene más de un nivel de profundidad. Por estructuras de un nivel simple que no tienen hijos, el IRawElementProviderSimple can puede ser usado en su lugar. El nodo de raíz del fragmento debe soportar la Interfaz IRawElementProviderFragmentRoot , que es derivada de este y tiene algunos métodos adicionales.

IRawElementProviderFragmentRoot

El elemento de raíz en un fragmento de UI debe soportar esta interfase. Otros elementos en el mismo fragmento requieren soporte de la Interfaz y



845

IRawElementProviderFragment

Implementado sobre el elemento de raíz de un fragmento UI para permitirle ser notificado de cuando es requerido elevar los eventos de automatización.

IRawElementProviderFragmentRootAdviseEvents

Implementado por los proveedores que quieren suministrar información acerca o quieren reponer los elementos con base en el HWND-based contenidos.

IRawElementProviderHwndOverride

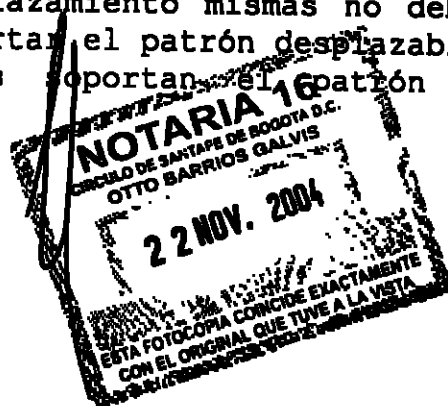
Implementado por proveedores que quieren suministrar información acerca de o quieren reponer los elementos con base en el ned HWND contenido.

IRawElementProviderOverrideType

La Interfaz de proveedor de UIAutomation, implementada por proveedores que quieren exponer propiedades para un elemento único. Para exponer propiedades y estructuras para más de un elemento único, ver la Interfaz derivada y **IRawElementProviderFragment** .e

IRawElementProviderSimple

El patrón de enrollado expone una habilidad del control para cambiar la porción de su region visible que está visible al Usuario por medio de enrollar o desplazar el contenido de enrollar o desplazar su contenido. Ejemplos: vistas de árbol, cajas de lista y otros contenedores que mantienen un área de contenido más grande que la región visible del control. Note que las barras de desplazamiento mismas no deben soportar el patrón desplazable; ellas soportan el patrón de

IScrollInteropProvider

ISelectionInteropProvider

RangeValue. Los servidores deben normalizar el desplazamiento (0 a 100). Esta Interfaz pública representa elementos UI que desplazan su contenido.

Suministra acceso las propiedades implementadas por proveedores que no son "de cuernos largos" en contenedores que manejan selección.

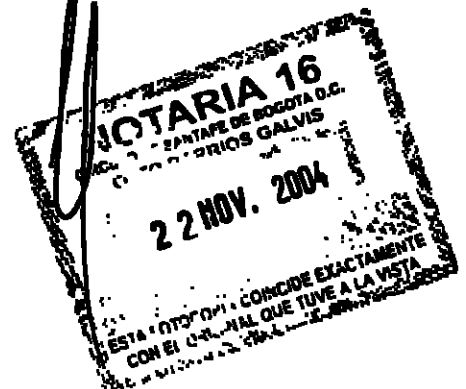
Suministra acceso a métodos y propiedades que definen y manipulan elementos seleccionados en un contenedor. Es solamente soportado sobre elementos lógicos que son hijos de elementos que soportan y ISelectionInteropProvider and y que son en si mismo seleccionables.

Expone una clasificación actual del contenedor y permite a los clients reclasificar programáticamente sus elementos. Algunos contenedores o carpetas mantienen el orden de clasificación cuando se insertan nuevos elementos., o mueven sus contenidos con base en contenidos actualizados (ejemplo: cajas de lista clasificadas Win32). Otros contenedores son solamente capaces de llevar a cabo clasificación de una sola vez lo que los vuelve fuera de fecha cuando los nuevos elementos son insertados (ejemplo: Excel).

ISortInteropProvider

Expone la habilidad para dividir un área de contenido de elementos en múltiples áreas o áreas interactivas.

ISplitInteropProvider



817

ITableInteropProvider

Identifica una rejilla que tiene información de encabezamiento.

ITableItemInteropProvider

Usado para exponer los elementos de rejilla con información de encabezamiento.

ITextInteropProvid

El patrón de texto expone una habilidad de control para manipular texto. Ejemplos: controles TextBox RichEdit que controla otros contenedores que contienen texto y propiedades relacionadas con texto. Esta Interfaz representa elementos UI que mantienen el texto.

ITextPointerInteropProvider

El patrón Text expone una habilidad de control para manipular texto. Ejemplos: el TextBox RichEdit controlan otros contenedores que contienen texto y propiedades relacionadas con texto. Esta Interfaz representa elementos UI que mantienen el texto.

IValueInteropProvider

Interfaz pública que representa elementos UI que están expresando un valor.

IVisualInformationInteropProvider

Usado para expresar información acerca de una imagen o una animación que conlleva información al Usuario.

IWindowInteropProvider

Expone la habilidad de elemento para cambiar su posición en pantalla o tamaño, lo mismo que cambiar el estado visual y cerrarlo.

IZoomInteropProvider

Expone el nivel de magnificación actual en un control y permite a los clientes cambiarlo programáticamente.

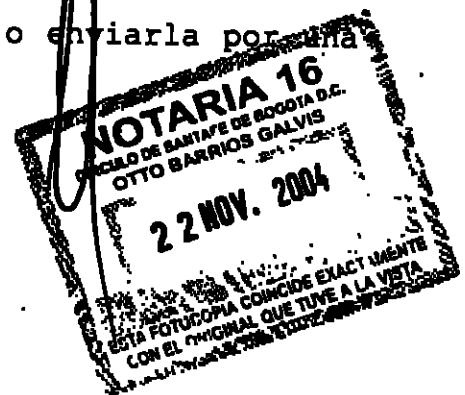


Enumeraciones

	Direcciones para
NavigateDirection	navegación con el
	árbol UIAutomation.
	Indica si este
	proveedor . está
	actuando como
RawElementProviderOverrideType	proveedor
	extralimitado o como
	un proveedor de area.

Espacio de Nombre de serialización

Un nombre de espacio de serialización 328 suministra un analizador que puede cargar o salvar una jerarquía de objetos (por ejemplo, elementos) de o en un archivo XML o un archivo con una representación binaria. Este proceso también establece propiedades asociadas con los objetos y manejadores de evento asociados. Adicionalmente, el nombre de espacio de serialización 328 suministra la funcionalidad para seriar y des-seriar objetos. La serialización es un proceso de tomar una representación de objetos en memoria viva, y producir una corriente de datos adecuados para guardar o salvarla en un dispositivo de almacenamiento (tal como un disco) o enviarla por una

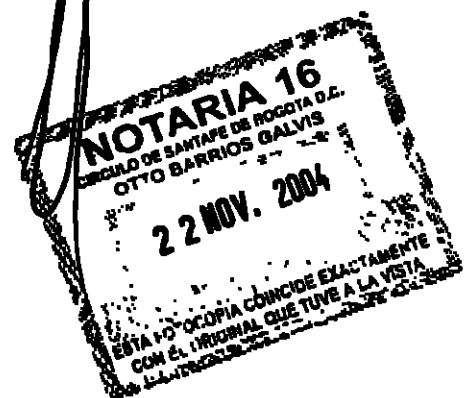


84

red. La deserialización es el proceso inverso; es decir, toma una corriente de datos y produce objetos de la corriente de datos. La deserialización también se refiere como a "analizar", y el código que implementa el análisis es conocido como el analizador.

Tanto la serialización como la deserialización soportan múltiples formatos de corrientes, tales como el formato XML referido como XAML o un formato binario que se llama BAML. Cuando el analizador es usado con un compilador de marcación, si puede generar "CAML", en donde la salida es código que genera en los objetos serializados.

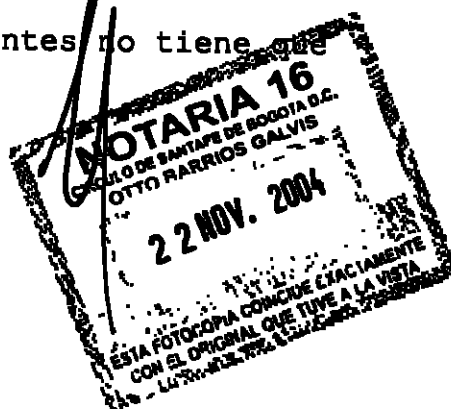
La serialización y la deserialización soportan rasgos de rango completo del XAML, incluyendo: sintaxis estándar, propiedades de compuesto, sintaxis compuesta compacta, sintaxis hijo implícita, sintaxis hijos explícitos, sintaxis de con tarjetas de colección explícita, y sintaxis de colección con tarjetas de colección implícita. Ambos la serialización y la deserialización son completamente extensibles. Las clases pueden definir como ellos desean ser serializados o



880

deserializados por medio de diversos mecanismos, tales como:

- Por defecto, los nombres de clase son tratados como nombres de tarjeta de marcación, y en las propiedades y eventos son tratados como atributos de marcación del mismo nombre.
- Las clases definen los convertidores de tipo (objetos auxiliares que se manejan desde `System.ComponentModel.TypeConverter`) para serializar o deserializar los valores de atributo de marcación.
- Las clases pueden soportar hijos implícitos por medio de implementar la Interfaz `IAddChild` (análisis) y usar atributos CLR para describir como serializar hijos implícitos.
- Las clases pueden controlar si una propiedad puede ser serializada/deserializada por medio de usar el `DesignerSerializationVisibility`, y controlar si es necesario serializar una propiedad sobre un objeto, particular por medio de usar el `DefaultValueAttribute` y `ShouldPersist methods`.
- O, un componente puede serilizarse/deserilizarse el mismo usando el contenido literal, en cuyo caso la forma serializada de los componentes no tiene que

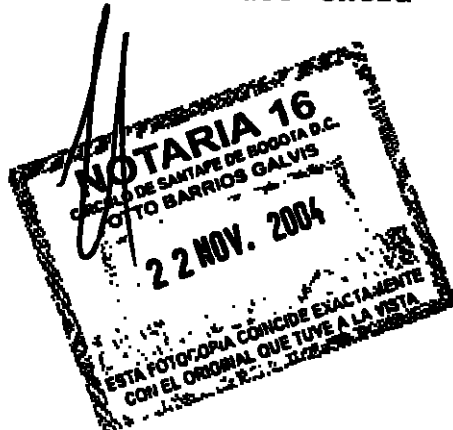


conformarse a la reglas del XAML (el componente requiere ser XML bien formado).

Aunque los anteriores ejemplos usan la palabra "marcación", los mismos conceptos amplían al formato binario (BAML).

Un segundo mecanismo de extensibilidad es "ganchos de diseñadores" que permiten que mostrar pieza de código case dentro del serializador y deserializador, e intercepte atributos durante el análisis y la serialización. Este mecanismo permite que nuevos rasgos de marcación sean creados, y permite un procesamiento adaptado de la sintaxis existente. El compilador de marcación usa ganchos de diseñadores para generar CAML.

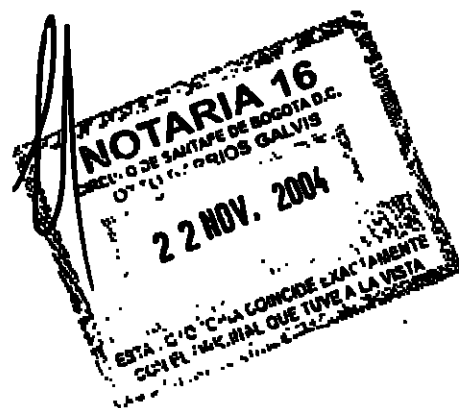
La serialización y la deserialización soporta una variedad de objeto (por ejemplo, cualquier clase de manejo desde System.Object). Las clases que se manejan desde System.Windows.DependencyObject pueden definir propiedades de dependencia, la cuales obtienen dos extra rasgos o características:



- La serialización puede contar automáticamente si una propiedad requiere ser serializada (no requiere para DefaultValueAttribute/ShouldPersist)
- El analizador puede llevar a cabo ciertas optimizaciones para largarse más rápido.

La serialización y la deserialización soporta espacios de nombre XML y archivos de definición que enmapean los espacios de nombre XML en espacios de nombre CLR. La clase de Mapeador permite que la definición o los mapeos arbitrarios de los espacios de nombres XML y de los nombres de tarjeta en clases CLR.

Los siguientes miembros ejemplo de la lista de tablas expuesta por el System.Windows.Serialization namespace.



883

Clases

CustomXamlDeserializer

La clase base par el ayudador o auxiliar de deserialización. Este debe ser sub-clasificado si el tiempo de diseño o de edición de la dirección de deserialización del analizador se requiere.

CustomXamlSerializer

La clase base par el auxiliar de serialización. Este debe ser sub-clasificado si el diseño o el tiempo de edición de la dirección de serialización del serializador se requiere.

DependencyPropertyContext

Información de contexto acerca de la propiedad actual que está siendo serializada. El XamlSerializer es responsable de colocar una instancia de esta clase en la pila de contexto antes de serializar cualquier propiedad sobre un objeto.

DPSerializationVisibilityAttribute

Especifica la visibilidad de esta propiedad o método según se ve por el serializador del diseñador.

EntityContext

Clase de base abstracta para el ObjectContext y elPropertyContext.

LayoutAttribute

Atributo para especificar el LayoutType para un rótulo.

Mapper

Maneja el Mapeado entre el Extensible Markup Language (XML) el Namespace Uniform Resource Identifier (URI) ay el Microsoft® .NET namespace esos tipos.

NamespaceMap

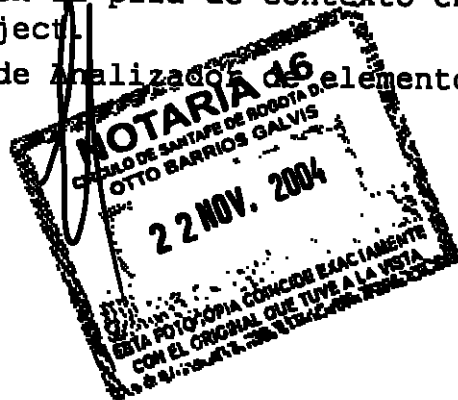
Contiene información del Mapeador que se usa para mapear entre un XML NamespaceUri y lo que el Ensamble, busca en, urtNamespace.

ObjectContext

Contexto de información acerca del objeto actual que esta siendo serializado. El XamlSerializer es responsable de colocar una instancia de esta clase en la pila de contexto en el SerializeObject.

Parser

Clase de Analizador de elemento



884

"Avalon" usado para crear un árbol "Avalon".

ParserContext

Suministra toda la información de contexto requerida por el Analizador.

PropertyContext

Información de contexto acerca de la propiedad actual que está siendo serializada. El XamlSerializer es responsable de colocar una instancia de esta clase en la pila de contexto antes de serializar cualquier propiedad sobre un objeto.

xtContext

Información de contexto acerca del objeto actual que esta siendo serializado. El XamlSerializer es responsable de colocar una instancia de esta clase sobre la pila de contexto en SerializeObject.

XamlAttributeNode

XamlAttributeNode

XamlClrArrayPropertyNode

XamlClrArrayPropertyNode

XamlClrComplexPropertyNode

EL XamlClrComplexPropertyNode, el cual es un objeto especificado como el hijo de un objeto

XamlClrEventNode

El XamlClrEventNode, el cual es un evento CLR sobre cualquier objeto. Note que este puede estar en una DependencyObject, o en cualquier otro tipo objeto.

XamlClrObjectNode

EL XamlClrObjectNode, el cual es cualquier otra cosa diferente a DependencyObject

XamlClrPropertyNode

El XamlClrPropertyNode, el cual es una propiedad CLR sobre cualquier objeto. Note que este puede estar sobre una DependencyObject, o cualquier otro tipo de objeto.

XamlClrPropertyParseException

Clase de excepción para la propiedad de analizador que analiza excepciones específicas.

XamlComplexDependencyPropertyNode

EL XamlComplexDependencyPropertyNode, es una DependencyProperty especificada como el hijo XML de un DependencyObject

XamlComplexPropertySerializer

El XamlComplexPropertySerializer es usado para serializar propiedades que



pueden no ser serializadas como atributos XML. ComplexProperties son en su lugar serializadas como marcaciones dentro del elemento.

XamlDefAttributeNode

XamlDefAttributeNode

XamlDefTagNode

XamlDefTagNode

XamlDependencyObjectSerializer

Se Extiende el XamlSerializer para incluir animaciones (específico Elemento).

XamlDependencyPropertyNode

El XamlDependencyPropertyNode, que es un DependencyProperty normal y simple sobre un DependencyObject

XamlDependencyPropertyParseException

Clase de excepción para analizador de propiedades dinámicas que analiza excepciones específicas.

XamlDesignerSerializerAttribute

Especifica el nombre tipo del serializador para la clase dada..

XamlDocumentNode

XamlDocument Node

XamlElementNode

El XamlElementNode, el cual representa un DependencyObject. Estos son diferentes de los objetos CLR porque ellos pueden tener Propiedades de Dependency asociadas con ellos.

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

Serializzala colección de nodos apuntando a por la propiedad. the node

XamlIListPropertyNode

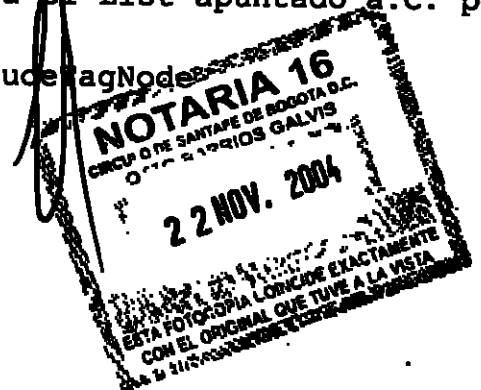
XamlIListPropertyNode

XamlIListSerializer

Serializa el List apuntado a.C. por la propiedad.

XamlIncludeTagNode

XamlIncludeTagNode



886

XamlLanguageNode
XamlLiteralContentNode

XamlNode

XamlParseException

XamlPIMappingNode

XamlResourceNode

XamlRootSerializer

 xmlRoutedEventNode

XamlSerializationCallbackException

XamlSerializer

XamlSerializerBase

XamlTextNode

XamlTextNodeSerializer

XamlXmlnsPropertyNode

XmlAttributes

XmlnsDictionary

XmlParserDefaults

XamlLanguageNode

XamlLiteralContentNode

Nodo de base en el cual todos los otros derivan.

Clase de excepción para el analizador de excepciones específicas.

El XamlPIMappingNode que mapea un espacio de nombre xml a un espacio de nombre clr y ensamble.

XamlResourceNode

El XamlSerializer es usado para persistir en el árbol lógico.

XamlRoutedEventNode

Clase de excepción para serialización de Excepciones específicas de evolución de llamada.

El XamlSerializer es usado para persistir en el árbol lógico.

Clase de base para el XamlSerializer proveyendo funciones auxiliares comunes.

XamlTextNode

Excede el XamlSerializer al caso especial case TextNodes.

XamlXmlnsPropertyNode

Una clase para encapsular atributos específicos XML-de un DependencyObject

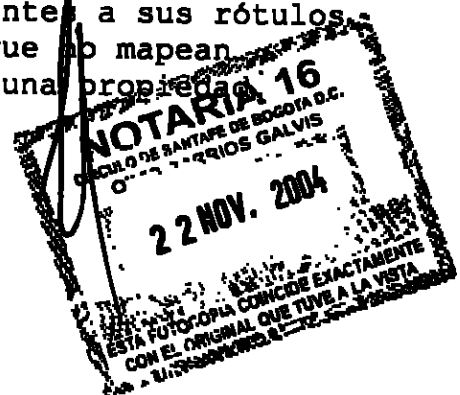
Un diccionario para controlar los mapeos de espacio de nombre XML.

Clase pública usada por Avalon.

Interfaces

IAddChild

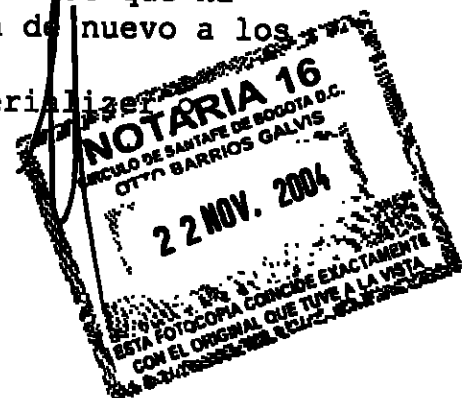
La Interfaz IAddChild es usada para analizar objetos que permiten que los objetos o textos subyacente a sus rótulos en marcación que no mapean directamente a una propiedad.



IBamlSerialize	La Interfaz de serialización para propiedades dinámicas escritas en archivos de BAML. Esta Interfaz suministra una propiedad booliana de solamente lectura llamada IsLoaded, un auxiliar de eventos Loaded, y métodos DeferLoad y EndDeferLoad.
ILoaded	
IPageConnector	Suministra métodos usados internamente por el BamlReader en contenido compilado.
IParseLiteralContent	Para el contenido literal-el contenido en el cual tiene supropio analizador y salvador de gravación.
IUriContext	La Interfaz IUriContext permite que elementos (como cuadro, y PageViewer) y convertidores tipo (como ImageData TypeConverters) una manera de asegurar que la base URI está establecida en ellos por el analizador, codegen para xaml, baml y caml como casos. Los elementos pueden entonces usar esta base uri para navegar.

Enumeraciones

LayoutType	Los tipos de Layout Types que pueden estar asociados con un objeto.
SerializationAction	Describe la acción del serializador o del deserializador es para ser tomada después de que ha sido llamada de nuevo a los auxiliares CustomXamlSerializers.



888

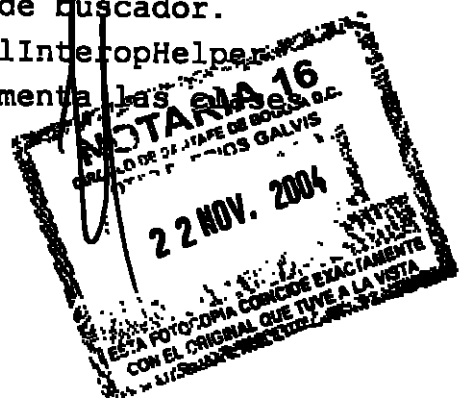
	CustomXamlDeserializer
	Describe la acción del serializador o
SerializationErrorAction	deserializador que es tomada cuando un error ha sido reportado.
XamlNodeType	Identificador para el XamlNodes.

Nombre de espacio de Interoperación

Un nombre de espacio interop 330 suministra un conjunto de clases que permite la interoperabilidad con otros sistemas operativo o plataformas de cómputo. Los siguientes miembros ejemplos de lista de tablas expuestas por el System.Windows.Interop namespace.

Clases

	La carpeta
	MarshalByRefObject sobre la clase Application para permitir la interacción con el objeto Application a través de AppDomains y para permitir la creación de Application en un hilo diferente.
ApplicationProxy	
	Clase interoperativa usada para implementar la parte administrada de un servidor Interop para hospedaje de buscador.
DocobjHost	
PresentationInteropHelper	VisualInteropHelper
WindowInteropHelper	Implementa las



WindowInteropHelper Avalon,
 las cuales ayudan la
 interop por/con heredar y
 Window.Avalon

Delegados

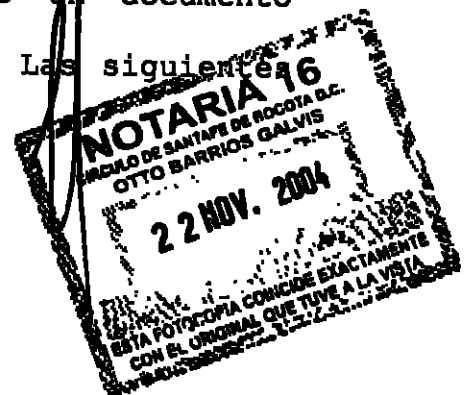
AppEntryPoint Delegado para la aplicación del
 método de punto de entrada de códigos.

Forms.Interop Namespace

Una forms.interop namespace 332 suministra un elemento que le permite a una aplicación hospedar una forma de operación de control.

System.IO.CompoundFile Namespace

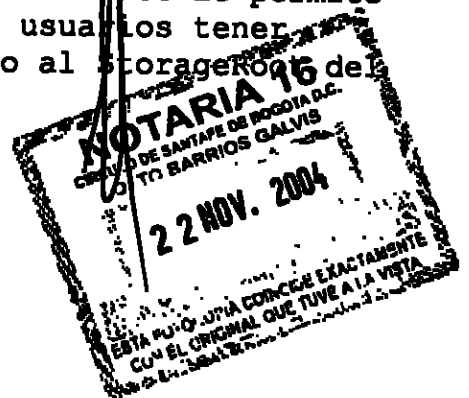
Otro namespace, System.IO.CompoundFile (no mostrado en la figura 3) suministran servicios para utilizar un archivo compuesto en el cual varios archivos distribuidos en documentos son almacenados. Estos servicios permiten la encriptación y compresión del contenido. Los servicios también soportan el almacenamiento de rendiciones múltiples del mismo contenido, tal como un documento refluible y un documento fijo en formato. Las siguientes



tablas listan ejemplos de miembros expuestos por el `System.IO.CompoundFile` namespace.

Clases

<code>CertificateHelper</code>	Auxiliar para obtener un certificado digital para firmar.
<code>CompoundFileByteRangeReference</code>	Referencia de subcorriente de componente que se refiere a un rango de bytes dentro de una corriente de archivos compuestos.
<code>CompoundFileIndexReference</code>	Referencia de subcorrientes de componente que se refiere a una entrada lógica dentro de una corriente de archivos compuestos.
<code>CompoundFileReference</code>	Una referencia a una porción de un archivo compuesto.
<code>CompoundFileReferenceCollection</code>	Una colección de solo lectura de referencias de archivos compuestos.
<code>CompoundFileStorageReference</code>	Referencia lógica a un almacenamiento de contenedor.
<code>CompoundFileStreamReference</code>	Referencia lógica a una corriente de contenedor.
<code>CompressionTransform</code>	<code>CompressionTransform</code> para uso en <code>DataSpaces</code> de archivos compuestos.
<code>ContainerHelper</code>	Esa clase le permite a los usuarios tener acceso al <code>StorageRoot</code> de



891

	contenedor actual con base en la instancia de aplicación cargada. Usado para manipular los espacios de datos dentro de una instancia específica del contenedor "Avalon" Esto es los datos que transforman a los módulos son enchufados en el contenedor para permitir rasgos similares a compresión de datos y a encriptación de datos. La clase de solo lectura que permite a los clientes inspeccionar y validar sus firmas digitales existentes. Una colección de solo lectura de firmas digitales. Esta clase es usada para crear, persistir, y manipular firmas digitales en un archivo compuesto. Contiene elementos de propiedades correspondientes a las propiedades en el documento OLE estándar en el conjunto de propiedades en el resumen de información. Clase para manipular objetos de versión. La excepción que es lanzada cuando el formato del DrmTransformInstanceData en el disco es inválido.
DataSpaceManager	
DigitalSignature	
DigitalSignatureCollection	
DigitalSignatureManager	
DocumentSummaryInfo	
FormatVersion	
InstanceDataFormatException	
OleProperty	



220

PropSet

RenditionInfo

RenditionInfoCollection

RenditionManager

RightsManagementEncryptionTransform

StorageInfo

StorageRoot

StreamInfo

SummaryInfo

TransformEnvironment

Clase para manipular información de una rendición. Esta clase también suministra API para agregar y remover almacenamientos y corrientes hacia y desde la rendición.

Una colección fuertemente escrita de objetos RenditionInfo.

Clase para manipular las rendiciones en un contenedor.

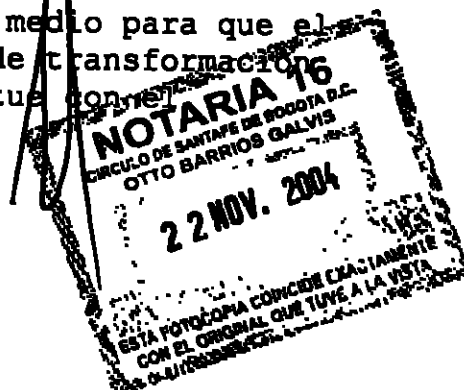
Clase que implementa la Interfaz IDataTransform, de modo que puede ser usada como una transformación de datos de contenedor que se implementa o que implementa la encriptación y desencriptación RM del contenido.

Clase para manipular almacenamientos en el archivo contenedor.

Representa en la clase contenedora principal. Esto es una instancia del StorageRoot por archivo compuesto.

Suministra accesos para manipular corrientes en el archivo contenedor.

Una instancia de esta clase es dada a cada objeto de transformación como un medio para que el objeto de transformación interactúe con el



893

ambiente suministrado por el administrador de espacio de datos. Esto no es mandatario de que un objeto de transformación mantenga una referencia sobre el objeto TransformEnvironment dado como puede escogerse para descartar sino hay necesidad de interactuar con el ambiente de transformación.

TransformInitializationEventArgs

Clase pública para pasar argumentos a los manejadores de evento.

UseLicenseInfo

Contiene información que describe una licencia de uso retornada del servidor Tungsten.

UseLicenseInfoEnumerator

Representa un enumerador sobre las licencias de uso almacenadas en los datos de instancia de la transformada DRM.

VersionTuple

Clase para una versión de tupla que consiste de números mayores y menores.

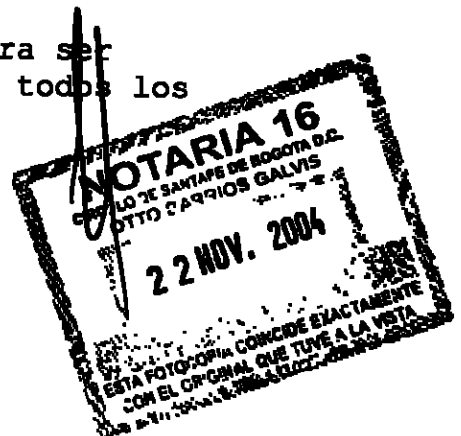
XmlDigitalSignatureProcessor

Procesador de firma usado por el administrador de firma digital para firmar y validar datos de acuerdo con la recomendación W3C.

Interfaces

IDataTransform

Interfaz para ser implementada por todos los



894

objetos de transformación de datos.

ILicenseStorage

Esta Interfaz es usada para desacoplar el RMTransform y el RMWizard por medio de construir implementaciones adaptadas de esta Interfaz se permite que terceras partes tomen ventaja del RMWizard sin forzarlos a usar el RMTransform.

ISignatureProcessor

Interfaz de procesador de firma.

IUnknownRCW

Esta Interfaz es usada para manejar opacamente cualquier Interfaz COM (archivada en la carpeta Runtime COM) como un IUnknown.

Enumeraciones

DigitalSignatureProcessor

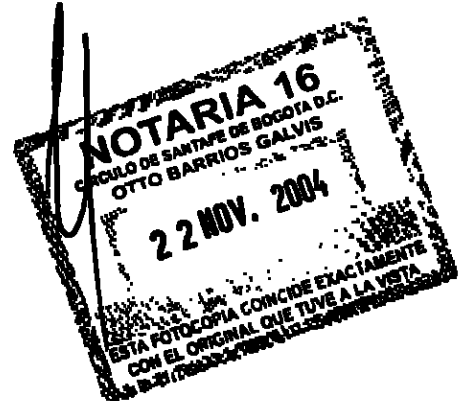
Procesadores de firma digital predefinidos.

SignatureProcessorNameType

Tipo de procesador de firma.

TransformIdentifierTypes

Cuando se nombra un objeto de transformación, la cadena que es pasada puede ser interpretada entre una de varias maneras. Este tipo enumerado es usado para especificar las semánticas de la cadena de identificación.

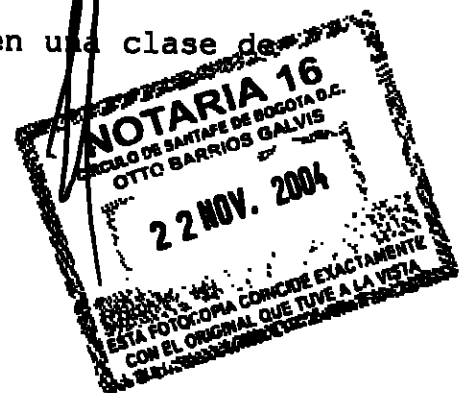


Delegados

InvalidSignatureHandler	Llamado con firmas que fallan su verificación completa.
TransformInitializeEventHandler	Método delegado para inicializar transformaciones.

System.Windows.Automation Namespace

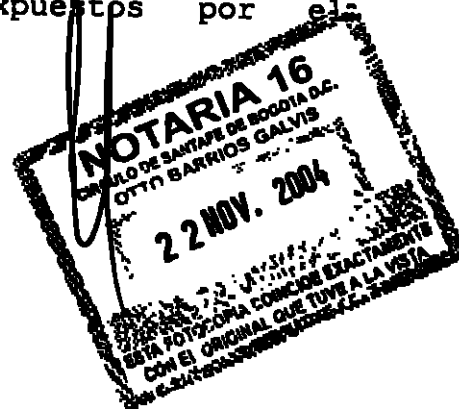
Un System.Windows.Automation namespace contiene miembros usados para soportar la accesabilidad y la automatización de la Interfaz de usuario. El sistema de accesabilidad incluye un lado de cliente y un lado de proveedor. Un sistema de herramientas incluye las herramientas de automatización de lado de cliente que incluyen una clase de automatización de cliente para buscar información de Interfaz de usuario. La clase de automatización de cliente incluye herramientas de registro de eventos y herramientas de descubrimientos de elementos lógicos. El conjunto de herramientas incluye adicionalmente herramientas de automatización de lado de proveedor para suministrar al cliente con información de la Interfaz de usuario. Las herramientas de automatización de lado de proveedor incluyen una clase de



396

automatización de proveedor que tiene herramientas para suministrar al cliente con información de eventos.

Una clase de automatización de cliente suministra métodos de automatización UI para uno o más clientes. La clase de automatización de cliente contiene métodos que no son específicos a ningún elemento UI. La clase de automatización de cliente puede suministrar un método para obtener un elemento lógico o directo desde un punto, manejar un manejo de ventana, o un elemento de raíz de escritorio. La clase de automatización de cliente adicionalmente puede suministrar métodos para encontrar un elemento lógico con base en criterios de entrada. La clase de automatización de cliente es preferiblemente e incluye también un método para registrar o desregistrar para notificaciones de evento. La clase de automatización preferiblemente también suministra funciones auxiliares para cargar los DLL Proxy, recuperar los nombres de propiedades localizadas y los patrones de control y llevar a cabo comparaciones de elementos. La clase de automatización de cliente también incluye métodos para que los clientes escuchen los eventos. Las siguientes tablas listan miembros ejemplo expuestos por el System.Windows.Automation namespace.



897

Clases

ApplicationWindowPattern

Expone el comportamiento y la información asociada típicamente con la ventana de aplicación de nivel superior. Los clientes pueden usar esta clase para arreglar en fila o en cascada las Interfaces de documento múltiple de las aplicaciones (MDI) hijo, encontrar su botón en la barra de tareas, y localizar secciones bien conocidas de su Interfaz de usuario tales como barras de herramientas y menús.

AssociatedInformationPattern

Expone la semántica y los metadatos para elementos UI que representan otros objetos.

Automation

Contiene métodos de automatización de Interfaz de usuario (UI) para clientes (tecnología de ayuda o manuscrito de prueba automatizado). Estos métodos no son específicos a un elemento particular.

AutomationEvent

No usar. Esta clase será removida en versiones futuras.

AutomationEventArgs

Evento de patrón o de adaptación de clase de argumentos.

AutomationFocusChangedEventArgs

Clase de argumentos de eventos de foco.

AutomationIdentifier

Clase base para identificadores de objeto con base en identidad. Esta clase es efectivamente abstracta, solamente clases derivadas son mencionadas.

AutomationIdentifierProxy

Clase interna usada para deserializar AutomationIdentifiers.



AutomationPattern

usarse directamente.

No usar. Esta clase será removida en versiones futuras.

AutomationPermission

Suministra un conjunto de permisos para acceder elementos UI. Esta clase no puede ser heredada.

AutomationPermissionAttribute

Suministra métodos y propiedades para el AutomationPermissionAttribute. Esta clase no puede ser heredada.

AutomationProperty

No usar. Esta clase será removida en versiones futuras.

AutomationPropertyChangedEventArgs

Clase de argumentos de eventos de PropertyChanged event

AutomationTextAttribute

Identificador para los atributos de automatización de texto.

AutomationTextPointer

Representa la posición de un carácter dentro del texto. El AutomationTextPointer suministra métodos y propiedades para tener acceso a texto y navegar en el texto.

AutomationTextRange

Usado para obtener, establecer, agregar y remover selección.

AutomationTextSelection

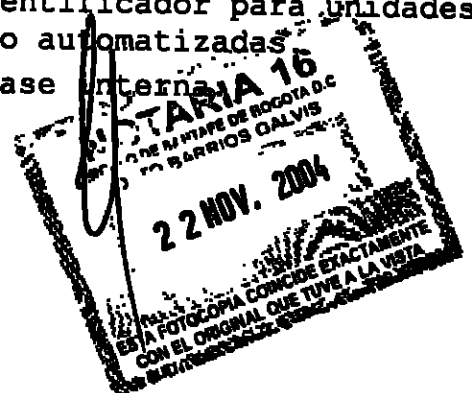
Propósito: el objeto AutomationTextSelection maneja toda la administración de selección de texto. La selección que el apuntador de inserción está sobre es la selección activa. Ejemplos de uso: es usado cuando el cliente quiere agregar, remover, o modificar o establecer la selección. Los clientes también encuentran que se ha seleccionado actualmente por medio del AutomationTextSelection.

AutomationTextUnit

Identificador para unidades de texto automatizadas

BasePattern

Clase interna



DockPattern

Expone una habilidad del elemento para cambiar su estado de archivo en tiempo de corrida.

ElementNotEnabledException

Esta excepción es lanzada cuando el código de cliente intenta manipular un elemento o controlar el elemento que no está habilitado actualmente.

ElementPath

El ElementPath suministra criterios necesarios para regresar o retornar subsecuentemente al elemento lógico previamente almacenado o gravado, modificado, o completamente creado por el suministrador o el proveedor de la aplicación.

ExpandCollapsePattern

Expone una habilidad del control para expandir para visualizar más contenido o para colapsar para esconder el contenido. Ejemplos incluyen botones de menú, botones de inicio, elemento de vista de árbol en Windows Explorer, y cajas combo.

GridItemPattern

Expone los elementos de una rejilla, le permite a los clientes rápidamente determinar las coordenadas de un elemento de rejilla.

GridPattern

Expone la funcionalidad básica de rejilla, tamaño, y navegación a celdas especificadas.

HierarchyItemPattern

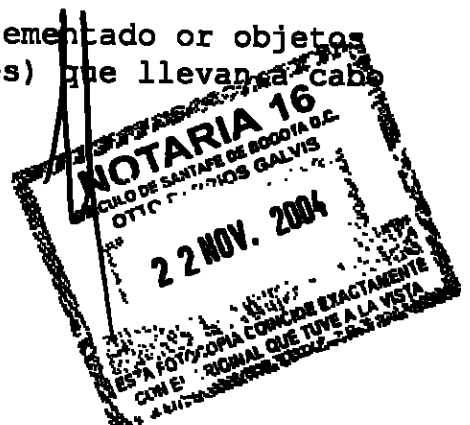
Expone la relación de jerarquía entre elementos de Interfaz de usuario del control independientes de su relación en el árbol lógico.

Input

Suministra métodos para enviar entradas de ratón o de teclado.

InvokePattern

Implementado or objetos (controles) que llevan a cabo



400

una acción única y no ambigua. La mayoría de estos controles no mantiene el estado; invocarlos inicia una acción en la aplicación. Ejemplos de controles que implementan esta Interfaz incluyen los botones expulsadores, los botones de barras de herramientas, los elementos de menú, hiperenlaces, cajas de verificación, botines e radio, el símbolo más en el control de vista de árbol, y los elementos en la vista de lista en el Windows Explorer.

Representa un elemento UI en el árbol lógico sin importar la implementación ("Avalon", , Microsoft® Win32®).

LogicalElement

Clase de argumentos de eventos cambiados en estructura lógica.

LogicalStructureChangedEventArgs

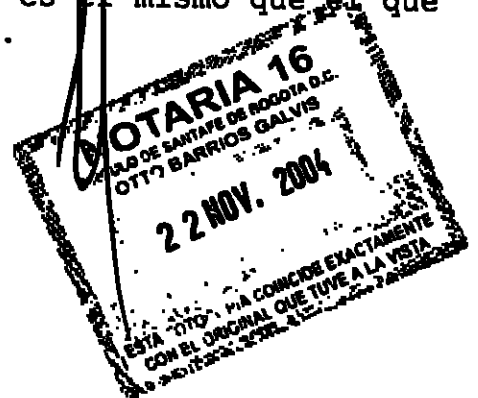
Expone una habilidad de los elementos para cambiar entre múltiples representaciones del mismo conjunto de información, datos, o hijos. Ejemplos incluyen la vista ListView (Miniaturas, Filas, Iconos, Listas, Detalles) Cartas Excel (ponques, Líneas, Barras, valores de celda con una fórmula), y Outlook y Calendario (Año, Mes, Semana, Día).

MultipleViewPattern

La excepción que es lanzada cuando el valor de un argumento está por fuera del rango permitido de valores, según se definió por el

NoClickablePointException

GetClickablePoint. Por ejemplo, cuando un rectángulo de límite está vacío, no tiene ancho o alto o el LogicalElement en ese punto no es el mismo que el que lo llamó.



ProxyAssemblyNotLoadedException

Esta excepción es lanzada cuando hay un problema al cargar un conjunto Proxy. Esto puede suceder en respuesta al Automation.RegisterProxyAssembly o cuando recarga los proxies por defecto cuando el primer LogicalElement de base HWND se encuentra.

RangeValuePattern

Expone una habilidad de control para manejar un valor dentro de un rango finito. Esto lleva a unos valores válidos mínimos y máximos del control y valor actual.

RawElement

Representa un elemento en el árbol de elementos directos.

ScrollPattern

Representa elementos UI que están expresando un valor.

SelectedItemPattern

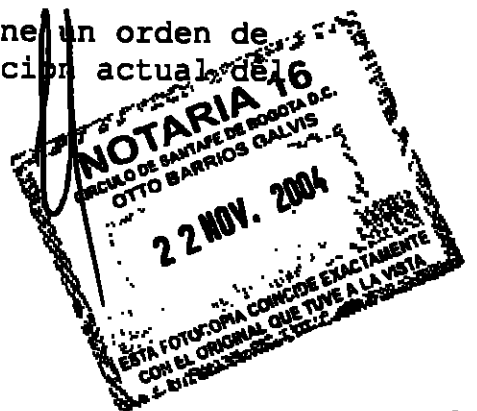
Representa elementos individuales en contenedores que manejan selección de clientes de automatización UI que usan esta clase para obtener información acerca de, o para manipular, elementos UI que soportan el patrón de control SelectionItemPattern. Ejemplos incluyen cualesquier elementos que puedan ser seleccionados en un control que soporta el patrón de control SelectionPattern tales como un elemento en una caja de lista o en la vista de árbol.

SelectionPattern

Representa contenedores que manejan selección. Usado por los clientes de automatización UI para obtener información acerca de, o para manipular los elementos de Interfaz de usuario que soportan el patrón de control SelectionPattern

SortPattern

Expone un orden de clasificación actual de



200
901

contenedor y permiten a los clientes elementos de reclasificación programáticamente.

SplitPattern

Representa Ventanas que pueden clonarse así mismas creando una ventana adyacente.

TableItemPattern

Representa elementos de rejilla con información de encabezamiento.

TablePattern

Representa una rejilla que tiene información de encabezamiento.

TextPattern

Representa texto tal como un control de edición.

TopLevelWindowEventArgs

Clase de argumentos de evento TopLevelWindowEventArgs

TreeLoadEventArgs

Clase de argumentos del evento TreeLoadEventArgs.

ValuePattern

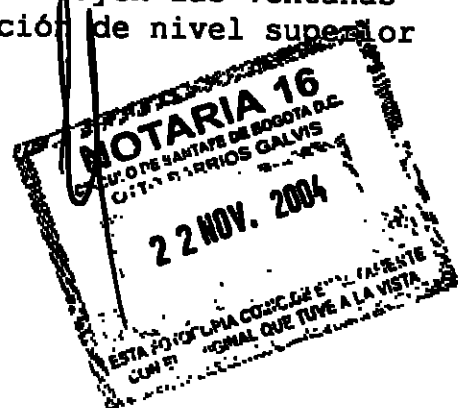
Expone valores para controles que se esparcen por un rango tales como los botones de radio, los botones en dos posiciones, las cajas de verificación, las cajas de edición, los valores de color RGB y los elementos de menú verificables.

VisualInformationPattern

Expone información acerca de una imagen o una animación que lleva información al usuario tal como las banderas de oscilación de Internet Explorer o los globos que giran que indican que un documento está siendo descargado o los papeles voladores de Windows Explorer que indican que el copiado está en progreso.

WindowPattern

Expone una habilidad del elemento para cambiar su posición en pantalla o tamaño. Ejemplos incluyen las ventanas de aplicación de nivel superior



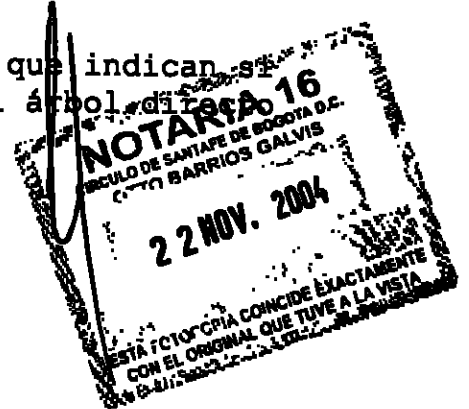
(Word o Windows Explorer), las ventanas pequeñas principales de Outlook (carpetas y mensajes de correo electrónico, tareas), y las ventanas de hijos MDI.

Expone el nivel de ampliación actual en un control y le permite a los clientes cambiarlo programáticamente.

ZoomPattern

Enumeraciones

<u>AutomationPermissionFlag</u>	Contiene las banderas de acceso para un objeto <u>AutomationPermission</u>
<u>ChildArrangement</u>	Diferentes maneras en que los hijos pueden ser ordenados.
<u>DockPosition</u>	El borde del contenedor que la ventana archivable le colgará.
<u>ExpandCollapseState</u>	Usado por el patrón ExpandCollapse para indicar un estado expandido/colapsado.
<u>GetDataOptions</u>	Opciones para el GetData
<u>HierarchyItemState</u>	Enumeración que indica el estado de un elemento de jerarquía: colapsado, expandido, o parcialmente expandido.
<u>ItemCheckState</u>	Enumeración que indica el estado de la caja de verificación, de los botones de radio, y controles similares.
<u>LoadState</u>	Árbol de banderas de estado.
<u>LogicalMapping</u>	Valores que indican un nodo en el árbol.



904

puede ser desplegado en el árbol lógico.

ProxyFlags

Enumeración usada para indicar resultados de solicitar una propiedad.

RowOrColumnMajor

Es el dato en esta tabla que mejor se presenta por filas o columnas.

ScopeFlags

Banderas usadas para definir el alcance cuando se escucha a eventos.

ScrollAmount

Usado por el ScrollPattern para indicar dirección (hacia delante o hacia atrás) y cantidad (página o línea) por medio del cual se ha desplazado.

SendMouseInputFlags

Banderas para el SendMouseInput. Estas banderas indican si el movimiento tuvo lugar, o si los botones fueron presionados o liberados.

SplitStyles

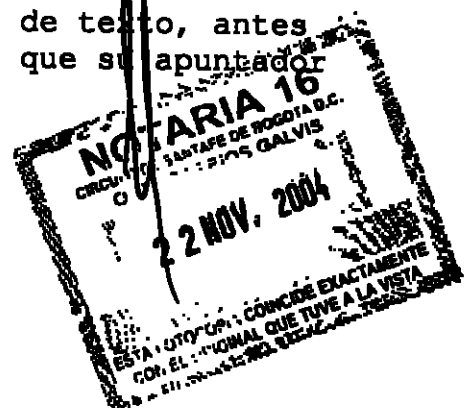
Bandera para las direcciones en las cuales una ventana se puede dividir.

StructureChangeType

Banderas que indican cambios en la estructura del árbol lógico.

TextPointerGravity

La gravedad determina la colocación de un AutomationTextPointer cuando el texto es insertado en el AutomationTextPointer. La gravedad es siempre desde el punto de vista de los apuntadores de texto, es decir yo estoy antes de mi carácter o yo estoy después de mi carácter. Cuando el usuario inserta un carácter en su localización de apuntador de texto, antes significa que su apuntador



905

será colocado antes del nuevo carácter, después significa que el apuntador será colocado después del nuevo carácter.

WindowChangeType

Banderas que indican cambios en las ventanas de nivel superior.

WindowInteractionState

El estado actual de la ventana para la interacción con el usuario.

WindowVisualState

Estados usados para describir el estado visual de la ventana.

WindowVisualState sigue las definiciones Office y HTML WindowState.

Estructuras.

MatchCondition Especifica criterios para encontrar elementos usando FindLogicalElement o FindRawElement.

NameAndData Contiene nombre, datos como un objeto, y datos como una cadena.

ProxyDescription Estructura que contiene información acerca de un Proxy.

SortInformation Información usada para clasificar.

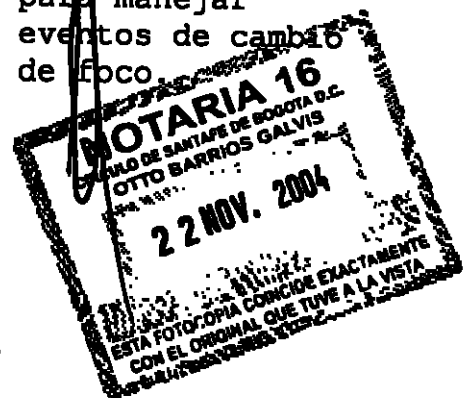
Delegados

AutomationEventHandler

Delegado para manejar AutomationEvents

AutomationFocusChangedEventHandler

Delegados para manejar eventos de cambio de foco.

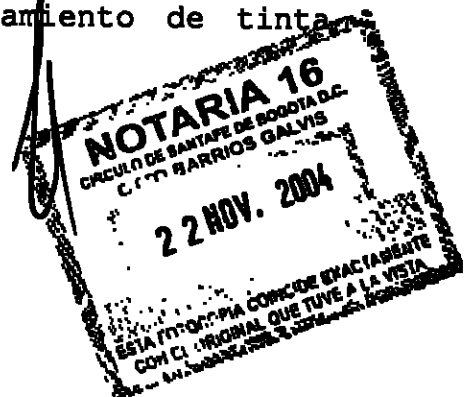


906

<u>AutomationPropertyChangedEventHandler</u>	Delegado para manejar eventos de cambios en la propiedad Automation
<u>LogicalStructureChangedEventHandler</u>	Delegados para manejar eventos de cambio en la estructura lógica.
<u>ProxyFactoryCallback</u>	Implementado por manejadores HWND; llamado por la estructura UI Automation para solicitar un Proxy para una ventana especificada. Debe regresar un Proxy si está soportado o un valor nulo sino está soportado.
<u>TopLevelWindowEventHandler</u>	Delegados para manejar eventos de ventana de nivel superior.

System.Windows.Ink Namespace

Un System.Windows.Ink namespace suministra clases que soportan sistemas de procesamiento de tinta electrónica. Las técnicas de procesamiento de tinta electrónica son útiles para una variedad de aplicaciones de software. Estas técnicas de procesamiento de tinta



90+

electrónica son particularmente aplicables al análisis de tinta electrónica, incluyendo análisis de diseños, clasificación, y reconocimiento de tinta electrónica. Ciertas técnicas de procesamiento de tinta electrónica permiten que la tinta electrónica sea procesada asincrónicamente con respecto a la operación de la aplicación de software implementando las técnicas, de modo que la tinta electrónica puede ser procesada sin detenerse o retrasar significativamente la operación de la aplicación de software. La aplicación de software puede continuar aún para aceptar la nueva entrada de tinta electrónica mientras que la entrada de tinta electrónica anterior está siendo procesada.

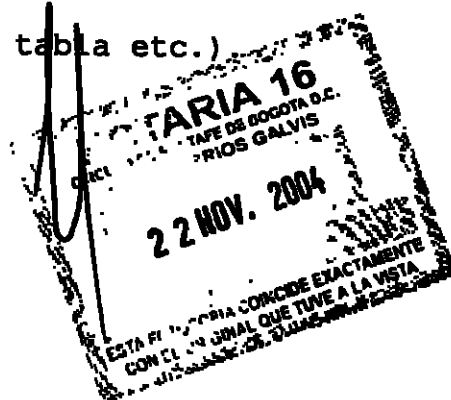
Una Interfaz de programación de aplicación cita un objeto analizador de tinta que recibe datos de documento para un documento que contiene un contenido de tinta electrónica desde una aplicación de software que hospeda el documento y corre sobre un primer hilo de proceso. El objeto analizador de tinta emplea entonces el primer hilo para hacer una copia de los datos del documento, suministra la copia de los datos del documento a un proceso de análisis de tinta electrónica, y regresa el control del primer hilo de procesamiento al proceso de



908

análisis. Después de que el proceso de análisis ha analizado la tinta electrónica, el objeto analizador de tinta reconcilia los resultados del proceso de análisis con los datos del documento actual para el documento.

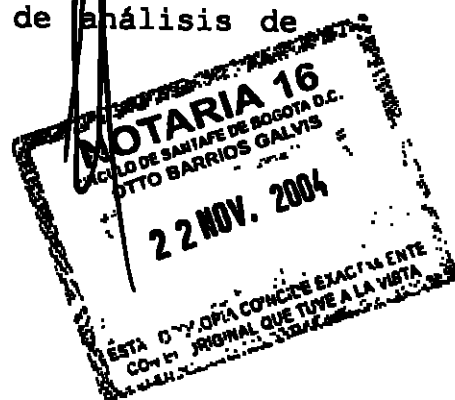
En modalidades particulares, los elementos en un archivo o documento pueden estar descritos, con base en su posición espacial relativa uno con respecto al otro. Por ejemplo tanto un golpe de tinta electrónica como un texto escrito a máquina o manuscrito pueden ser descritos en términos del mismo sistema de coordenadas espacial. Usando información espacial para describir los elementos de un documento, la aplicación de software que administra el documento puede mantener una estructura de datos que describe la relación entre sus elementos del documento. En particular, la aplicación de software puede mantener una estructura de datos que describe tanto la clase de los varios elementos del documento y define asociaciones entre los varios elementos de documento. Estas asociaciones pueden ser definidas, por ejemplo, como información usada para enlazar los datos del golpe de tinta electrónica o colecciones de ellos con otros elementos en el documento electrónico (tal como palabras, líneas, párrafos, dibujos, celdas de una tabla etc.)



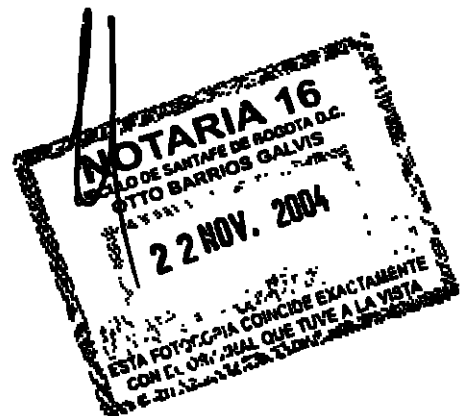
909

Por medio de describir sus elementos de documento en un archivo o en la estructura de datos del documento con base en su posición espacial, los elementos del documento para una variedad de tipos de archivo pueden emplear técnicas comunes para identificar y manipular sus elementos de documento. Más particularmente, una variedad de aplicaciones de software pueden describir elementos de documento dentro de un documento con base en su posición espacial y emplea esta posición espacial haciendo referencia al uso de métodos de análisis de tinta electrónica comunes. Adicionalmente, por medio de especificar una región particular de un documento para análisis, cada aplicación de software puede limitar el proceso de análisis a solamente los elementos deseados dentro de un documento.

Para analizar nuevas entradas de tinta electrónica en un documento, la aplicación de software que administra el documento modifica una estructura de datos asociada con el documento para incluir la nueva tinta que va a ser analizada. La aplicación de software entonces suministra esta estructura de datos (o porciones relevantes del mismo) a una herramienta de análisis de

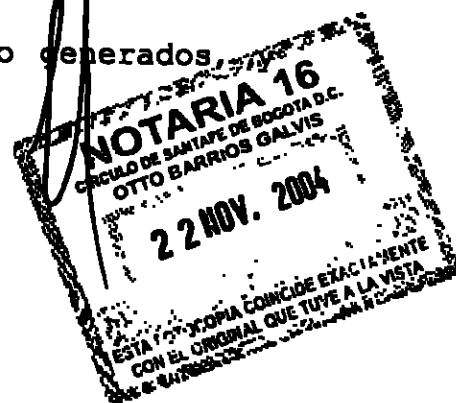


tinta, la cual copia algo o toda la estructura de datos para análisis (y opera sobre esta copia de datos que es independiente de la estructura de datos del documento del programa). La herramienta de análisis de tinta pasa la copia a un proceso de análisis tal como un proceso analítico (por ejemplo, un proceso de análisis de diseño y/o un proceso de clasificación). La aplicación de software puede retomar su operación normal, incluyendo recibir nueva entrada de tinta electrónica y/o otros datos, mientras el proceso o procesos de análisis de tinta está (están) siendo llevados a cabo. En adición a recibir nueva tinta electrónica, cualquier "otro dato" puede ser recibido por el programa de aplicación, por ejemplo, modificar datos de un tamaño, localización, o contenido de tinta, texto, imágenes, gráficos, tablas, diagramas de flujo, diagramas y similares existentes, datos que agregan texto adicional, imágenes, gráficos, tablas, diagramas de flujo, Diagramas y similares; datos que eliminan texto, imágenes, gráficos, tablas, diagramas de flujo, diagramas, y similares existentes. Después todos los procesos de análisis deseados que son completados, los resultados del análisis son regresados a la herramienta del análisis de tinta.



Una herramienta de análisis de tinta lleva a cabo una variedad de funciones para facilitar el procesamiento de la tinta electrónica. La herramienta de análisis de tinta puede ser implementada como una Interfaz de programación, tal como un API. Adicionalmente, la herramienta de análisis de tinta puede ser implementada como un grupo de rutinas de objeto de software y la información relacionada que puede ser llamada por una aplicación de software según se requiera para analizar tinta en un documento.

En una modalidad, un API que involucra una implementación de una herramienta de análisis de tinta (referida de aquí en adelante como un API de Análisis de tinta) puede contener dos clases de núcleo. La primera clase se refiere como la clase "Análisis de Contexto", y la segunda clase es la clase "Analizador de Tinta". Los componentes de la clase "Análisis de Contexto" son usadas para crear un objeto de Análisis de Contexto. Los componentes de la clase "Analizador de Tinta" son entonces usados para crear y emplear un objeto que suministra un documento independiente del objeto de contexto de análisis a un proceso de análisis, determina cuando los resultados de análisis han sido generados.



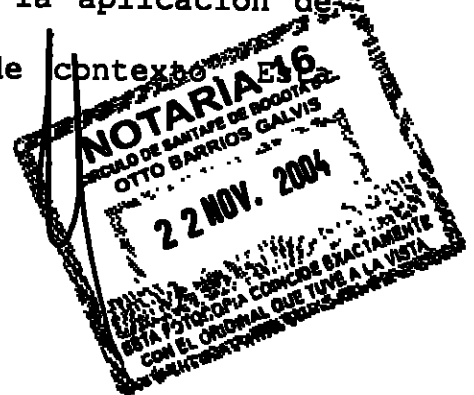
910

912

reconcilia los resultados del análisis con el estado actual de un documento.

La clase de análisis de contexto es implementada por una aplicación anfitrión para crear un objeto de contexto de análisis de contexto, que sirve como una vista Proxy en el árbol de documento interno de la aplicación de software. El objeto de análisis de contexto contiene todos los datos de tinta no analizados, y el objeto de análisis de contexto es usado para identificar que datos de tinta no analizados deben ser analizados. El objeto de análisis de contexto también contiene información acerca de la tinta analizada. Esta tinta analizada previamente puede ser usada para decidir como la tinta actualmente no analizada debe ser analizada y ella misma puede ser modificada en el curso del análisis de la tinta no analizada. Adicionalmente, el objeto de análisis de contenido contiene información acerca de contenido que no es tinta del documento, que es usado para clasificar apropiadamente la tinta como anotaciones en el contenido que no es tinta.

La clase de análisis de contexto incluye un constructor que, cuando es llamado por la aplicación de software crea el objeto de análisis de contexto.



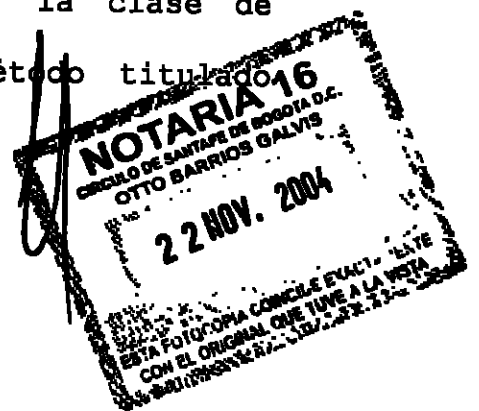
clase puede también incluir varias propiedades para el objeto de análisis de contexto, incluyendo la propiedad titulada "Región Sucis [get;]." La propiedad de Región Sucia define la porción del documento (y por lo tanto la porción del objeto de análisis de contexto) que contiene datos de tinta no analizados.

La clase de análisis de contexto también puede incluir una propiedad titulada "Rootnode {get;}", que identifica la parte más superior o el nodo de raíz del contexto en el objeto de análisis de contexto. Este nodo de raíz del contexto contiene, como nodos de contexto hijos todos los otros objetos de nodo de contexto para el objeto de análisis de contexto dado. En modalidades particulares, el nodo de raíz de contexto debe ser del tipo nodo de contexto "Root." En modalidades en donde la aplicación implementa su propio objeto de análisis de contexto, el objeto de análisis de contexto puede tener otros nodos de contexto como hermanos del nodo de raíz de contexto, pero los componentes de la clase analizador de tinta pueden estar limitados a considerar nodos de contexto contenidos por el nodo de raíz de contexto.



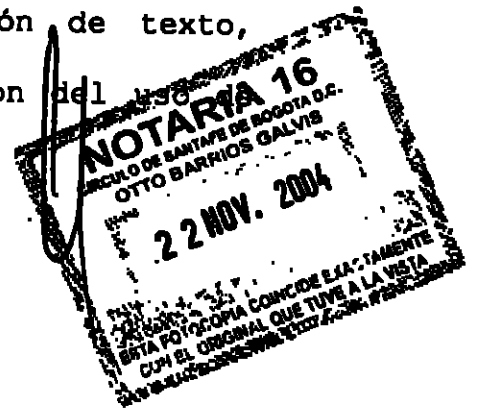
La clase de análisis de contexto puede adicionalmente incluir la propiedad "Analysis Hints {get;}," que regresa un arreglo o matriz de objetos de pistas de análisis establecido por la aplicación de software. Como será discutido en más detalle más adelante, los objetos de pistas de análisis pueden contener cualquier tipo de información que pueda ayudar al proceso de análisis. Esta información puede incluir, por ejemplo, como factoides, guías o una lista de palabras. Este también puede incluir información que establece un lenguaje que va a ser usado para el análisis, información que designa la tinta no analizada como texto manuscrito solamente o de dibujos solamente, o suministra cualquier clase de guía al proceso de análisis, tal como identificar la tinta como listas, tablas, formas, diagramas de flujo, conectores, contenedores y similares.

En adición a estas propiedades, en la clase de análisis de contexto puede incluir también varios métodos que pueden ser llamados por, por ejemplo, la aplicación de software para hacer que el objeto de análisis de contexto ejecute una tarea. Por ejemplo, la clase de análisis de contexto puede incluir un método titulado



"FindNode (Guid id)." Cada nodo en el objeto de análisis de contexto tiene un identificador globalmente único (o GUID), y este método localizará el nodo especificado en la llamada en cualquier parte en el objeto de análisis de contexto.

Tal como la clase de análisis de contexto, la clase de analizador de tinta también define un constructor público que permite a la aplicación de software que da una instancia de la clase (es decir, un objeto analizador de tinta), junto con varias propiedades, por ejemplo, este puede contener una propiedad titulada "User Interfaz Context {get; set;}," que define el hilo de procesamiento al cual los resultados del proceso de análisis son regresados. Esta propiedad permite que los resultados sean sincronizados con otro objeto. Por ejemplo, si este es establecido en la forma principal, los resultados del analizador serán disparados sobre el hilo principal de la aplicación. Esto también puede contener la propiedad "AnalysisOptions AnalysisOptions {get; set;}", que especifica diversos criterios que pueden ser usados para el proceso de análisis. Estos criterios pueden incluir, por ejemplo, reconocimiento de habilitación de texto, habilitación del uso de tablas, habilitación del uso de

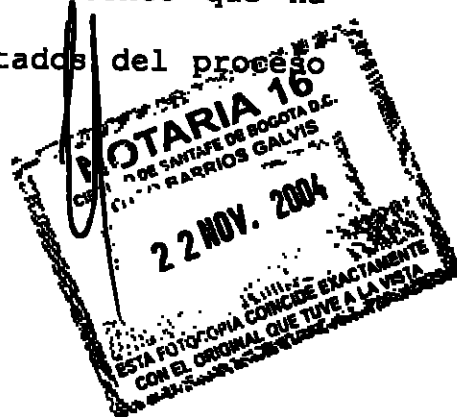


96

listas, habilitación del uso de anotaciones, y habilitación del uso de conectores y contenedores.

La clase de analizador de tinta puede incluir diversos métodos. Por ejemplo, esta clase puede incluir un método titulado "AnalysisRegion Analyze ()". Este método inicia un proceso de análisis sincrónico. Los datos del elemento del documento son pasados en este método, el cual describe el estado actual del documento e indica que tinta en el documento requiere ser analizada. Con algunas modalidades, los datos del elemento del documento pueden estar provistos como el objeto del análisis de contexto (es decir, AnalysisRegion Analyze (Analysis Context)), como se anotó anteriormente. Alternativamente, los golpes de tinta individuales pueden ser pasados al proceso de análisis, sea usando una referencia a los golpes (es decir, AnalysisRegion Analyze (Strokes)) o referenciado como una propiedad del objeto Analizador de Tinta (por ejemplo, InkAnalyzer.Strokes {get;set}) sin pasar ninguna propiedad al método Analyze.

Una vez que el proceso de análisis es completado, este método regresará una referencia al documento independiente de objeto de análisis de contexto que ha sido modificado para contener los resultados del proceso



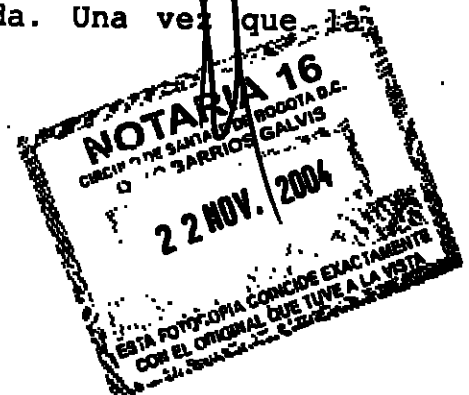
de análisis. El método también regresa un valor de `AnalysisRegion` que describe el área en el documento en donde los resultados han sido calculados.

En la clase de Analizador de Tinta también puede incluir un método titulado "`AnalysisRegion Analyze(AnalysisContext, waitRegion)`." Este método es el mismo de que el método sincrónico `Analysis Region Analyze()` discutido anteriormente, pero este solamente tiene tinta analizada si los resultados son requeridos en el área especificada `waitRegion`. Más particularmente, la llamada a este método identificará el objeto de análisis de contexto para el documento y la región del objeto de análisis de contexto (referido como el "`waitRegion`") para el cual el proceso de análisis debe analizar sincrónicamente. En ciertas modalidades, todas las otras regiones del objeto de análisis de contexto serán ignoradas, al menos que el proceso de análisis requiera analizar el contenido en aquellas regiones para poder completar su análisis del `waitRegion`. Como se discutió anteriormente, el objeto de análisis de contexto pasado con este método contiene una propiedad llamada el "`DirtyRegion`" que describe las áreas del documento que requieren análisis. Por medio de especificar una



waitRegion particular, la aplicación de software puede obtener los resultados de análisis más rápidamente para una región específica de interés en vez de tener todos los datos de tinta en el documento analizado. Cuando sea de estos métodos Analyze sean llamados, cada proceso de análisis disponible será ejecutado. También, debido a que estos métodos Analyze son llamados sincrónicos, no hay necesidad de ejecutar el proceso de reconciliación luego de su conclusión, ni tampoco disparar un evento una vez que ha sido completado.

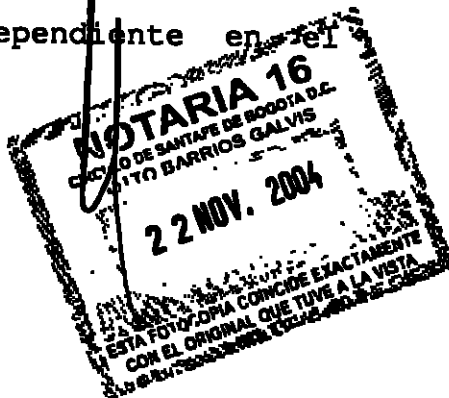
La clase de Analizador de Tinta puede también incluir un método titulado "BackgroundAnalyze(AnalysisContext)." Este método inicia la operación de análisis especificada, pero lo hace sobre un hilo de análisis de respaldo separado. Así, el método regresará el control al hilo de procesamiento principal casi inmediatamente mientras la operación de análisis real se completa en el respaldo. En particular, este método regresará un valor de "verdadero" si el proceso de análisis fue iniciado exitosamente. Nuevamente, el valor AnalysisContext pasado al método identifica el objeto de análisis de contexto para el documento e indica que tinta en el documento requiere ser analizada. Una vez que la



operación de análisis se ha completado en el hilo de respaldo, un evento surge para permitir que la aplicación de software tenga acceso a los resultados. El evento contiene los resultados y el método de reconciliación que es usado para incorporar los resultados de nuevo dentro del objeto de análisis de contexto para el estado actual del documento cuando los resultados son regresados.

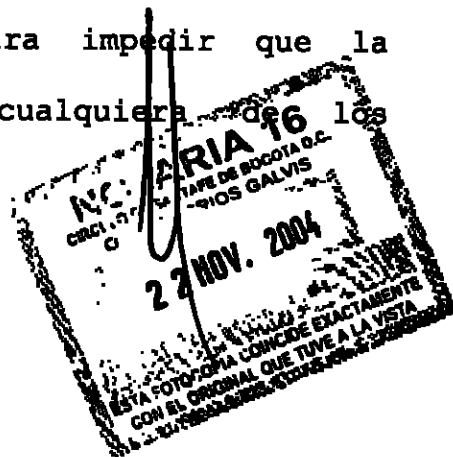
Cada uno de estos tres métodos Analyze a su vez llaman el método "Clone" en una clase "Analysis Region". Usando el método "Clone", estos métodos Analyze crean el objeto de análisis de contexto del documento independiente que es subsecuentemente modificado por el proceso de análisis para mostrar los resultados de análisis.

La clase Analizador de Tinta puede también incluir un método titulado "Reconcile (AnalysisContext current, AnalysisResultsEventArgs resultArgs)," el cual la aplicación de software llama después de recibir el evento de resultados que fue causado por llamar el método BackgroundAnalyze(AnalysisContext). El método Reconcile compara los resultados de análisis contenidos en el objeto de análisis de contexto independiente en el



documento con la versión actual del objeto de análisis de contexto mantenida por la aplicación de software. Este método identifica los nodos que requieren ser agregados y removidos de la versión actual del objeto de análisis de contexto, e identifica si cualquiera de las siguientes propiedades de un nodo existente ha cambiado: sus resultados de reconocimiento, su localización, los golpes de tinta asociados con el nodo, o cualquier otro dato asociado con los resultados de la operación de análisis. Este método también escribe estos cambios identificados en la versión actual del objeto de análisis de contexto. Este método es sensible al ordenamiento del ordenamiento de contexto de nodo tal como el orden de nodos de palabra de contexto sobre un nodo de línea de contexto.

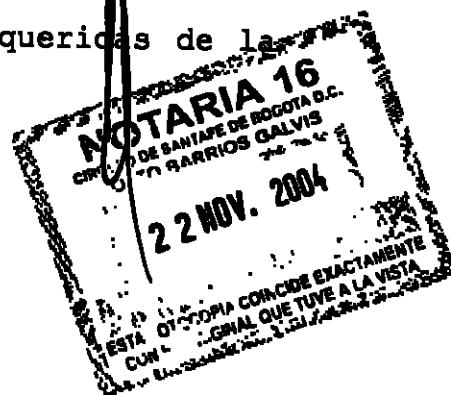
El resultado del análisis (esto es, el valor de la propiedad AnalysisResultsEventArgs) son pasados de nuevo con este método, a medida que ellos contienen una estructura de resultados pública y una estructura de resultados privada. La estructura pública es regresada de modo que la aplicación software pueda pre-visualizar los cambios que ocurrirán en la etapa de reconciliación. La estructura privada es incluida para impedir que la aplicación de software cambie cualquiera de los



44

resultados del análisis antes del proceso de reconciliación.

La clase Analizadora de Tinta puede incluir también los métodos titulados "Recognizers RecognizersPriority()" y "SetHighestPriorityRecognizer(recognizer)." Cuando la tinta requiere ser reconocida, el reconocedor apropiado será usado con base en el lenguaje y las capacidades. De acuerdo a lo anterior, el método Recognizers RecognizersPriority() regresa los procesos de reconocimiento en el orden en el cual ellos serán evaluados por el objeto Analizador de Tinta. El orden es determinado por el sistema dependiendo de los procesos de reconocimiento disponibles existentes, pero puede ser excedido por la aplicación de software por medio de llamar el SetHighestPriorityRecognizer(recognizer) En el objeto Analizador de Tinta. El InkAnalyzer enumerará por medio de esta lista ordenada hasta que un reconocedor apropiado pueda ser encontrado. El método SetHighestPriorityRecognizer(recognizer) eleva la prioridad de un proceso de reconocimiento. Por medio de llevar la prioridad de un proceso de reconocimiento particular, ese proceso de reconocimiento será usado si casa con el lenguaje y las capacidades requeridas de la



329

operación de reconocimiento actual. En esencia, el `SetHighestPriorityRecognizer(recognizer)` empuja el proceso de reconocimiento designado a la parte superior de la lista regresada por el método `RecognizersPriority`.

La clase Analizador de Tinta puede también contener un método titulado "`AnalysisRegion Abort()`," que puede usar un objeto de análisis de contexto como un parámetro. Este método permite que la operación de análisis anterior de respaldo sea terminada tempranamente. Este método regresa una región de análisis que describe el área que estaba siendo analizada antes del aborto. Así, si la aplicación de software intenta continuar la operación de análisis en un tiempo posterior, esta región puede fundirse dentro del `DirtyRegion` del objeto de análisis de contexto para el estado actual del documento. Todavía adicionalmente, la clase Analizador de Tinta puede incluir un evento titulado "`AnalysisResultsEventHandler`," el cual dispara al objeto `InkAnalyzer` tan frecuentemente como sea práctico. Más particularmente, este evento puede disparar entre procesos de análisis y por lo menos una vez cada cinco segundos. Este evento puede ser usado para suministrar a la aplicación con una actualización de modo



que el estado de un proceso de análisis asincrónico en curso (o procesos).

El Ink Analysis API puede también incluir clases en adición a la clase de Análisis de Contexto y la clase de Analizador de Tinta. Por ejemplo, el Ink Analysis API puede incluir una clase de Context Node. Esta clase puede incluir diversos componentes relacionados con los nodos de contexto que construyen el objeto de análisis de contexto y el objeto de análisis de contexto independiente del documento, tal como la propiedad titulada "ContextNodeType Type {get;}." Cada nodo de contexto tiene un tipo, y existe un conjunto específico de reglas al cual cada tipo debe adherirse. Esto incluye reglas tales como que tipo de nodos de contexto hijo son permitidos y si hay asociados directamente golpes al nodo de contexto o solamente vía sus nodos de contexto hijo.

Los tipos posibles de nodos de contexto pueden ser definidos en una enumeración ContextNodeTypes y puede incluir, por ejemplo, los siguientes tipos: Un nodo InkAnnotation, que representa la anotación de datos de tinta de datos que no son texto; un nodo InkDrawing que representa datos de tinta que forman un dibujo, un nodo



InkWord, que representa datos de tinta que forman una palabra, un nodo Line, que contiene uno o mas nodos InkWord y/o nodos TextWord para palabras que forman una línea de texto; el nodo ListItem, que puede contener nodos Paragraph, Image, o similares esperados en una lista; y un nodo List, que contiene uno o más nodos ListItem cada uno describiendo una entrada en la lista. Los tipos de nodo pueden también incluir un nodo NonInkDrawing, que representa una imagen de dibujo que no es tinta; un nodo Object, que representa datos no cubiertos por otros valores de la enumeración ContextNodeType; un nodo Paragraph, que contiene uno o más nodos Line correspondientes a las líneas que forman un párrafo; un nodo Picture o Imag, que representa una imagen de dibujo o foto; un nodo Root, que sirve como el nodo más superior en un objeto de análisis de contexto; un nodo Table, que contiene nodos que representan elementos que hacen una tabla; un nodo TextBox, que representa una caja de texto; un nodo TextWord; y un nodo UnclassifiedInk, correspondiente a datos de tinta que no han sido todavía clasificados. Los tipos de nodo también pueden incluir un nodo Group, para grupos de otros nodos, un nodo InkBullet para elementos de bala, un nodo Row para elementos de documentos presentados en una fila.



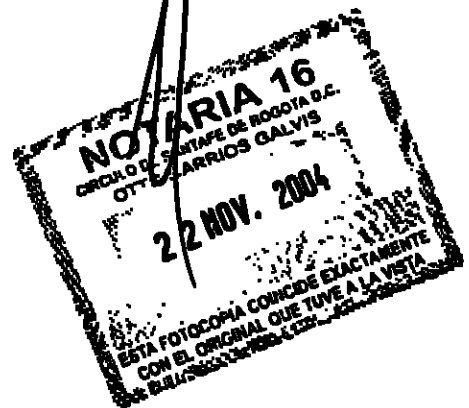
una tabla, y un nodo Cell para elementos de documentos presentados en una celda de una tabla.

La clase Context Node puede también incluir la propiedad titulada "GUID Id {get;}," la cual es un identificador globalmente único para el nodo de contexto actual. Para poder permitir al acceso a cualquier nodo de contexto deseado, cada nodo de contexto dentro de un objeto de análisis de contexto único debe tener un identificador único. Esta clase puede también incluir una propiedad titulada "AnalysisRegion Location {get;}," que identifica la localización en el espacio del documento en donde el nodo de contexto relevante está posicionado realmente. Un AnalysisRegion es una estructura bidimensional que agrupa uno o más posibles rectángulos no adyacentes similar a estructuras junta. Esta clase puede también incluir la propiedad titulada "StrokeCollection Strokes {get;}", que identifica los golpes de tinta asociados con el nodo de contexto relevante. Con modalidades particulares, solamente nodos de contexto de hoja (tales como nodos Word, Drawing a Bullet) son permitidos por el Ink Analysis API para que tenga golpes. La aplicación de software puede usar esta propiedad para referenciar los golpes en el nivel de nodo

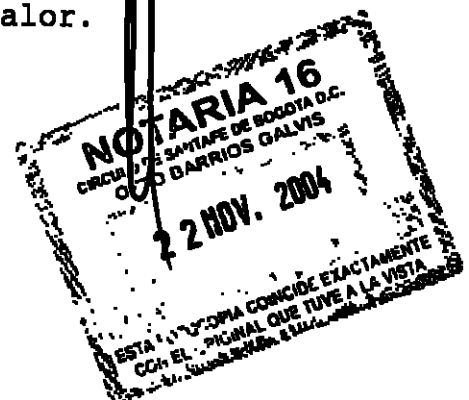


de hoja para todos los nodos de contexto ancestro (por ejemplo, el nodo raíz que contiene unas referencias de golpes a todos los golpes en el objeto de análisis de contexto relevante.)

Adicionalmente, esta clase puede incluir la propiedad titulada "ContextNode ParentNode {get;}," que identifica el nodo de contexto padre que contiene el nodo de contexto relevante. En modalidades particulares, los nodos son siempre creados para depender de un nodo de contexto padre, con el nodo de contexto Root siendo un miembro estático de un objeto de análisis de contexto. Esta clase puede también incluir la propiedad "ContextNode[] SubNodes {get;}" QUE identifica todo los nodos de contexto que son hijos directos del nodo de contexto relevante. Esto es, esta propiedad solamente identificará aquellos nodos de contexto hijo y que están un nivel abajo en el objeto de análisis de contexto. Por ejemplo, el valor de esta propiedad para un nodo de contexto Paragraph solamente identificará los nodos de línea de contexto contenidos por el nodo Paragraph, y no los nodos de palabras de contexto que son hijos del nodo de línea de contexto.



Esta clase puede también incluir la propiedad titulada "RecognitionResult RecognitionResult {get;}," que suministra el resultado de reconocimiento según se calculó por el proceso de análisis de reconocimiento relevante o procesos, y el RecognitionResult puede representar más de una línea de texto hasta más de un lenguaje. El Recognition Result está disponible para cada nodo de contexto en el documento del objeto de análisis de contexto independiente aunque la propiedad RecognitionData que es establecida por el proceso de análisis de reconocimiento y es usada para crear el objeto RecognitionResult puede solamente establecerse en un nivel del árbol del nodo de contexto para evitar la duplicación de datos. Si el nodo no tiene un RecognitionData asociado con este, este fundirá los resultados de reconocimiento de todos sus sub-nodos o extraerá el resultado de reconocimiento de sus padres. Esta clase puede también incluir la propiedad titulada "Stream RecognitionData {get; set;}," que es la forma persistente del valor RecognitionResult. Nuevamente, el proceso de análisis de reconocimiento produce el valor Stream RecognitionData que es establecido sobre el nodo de contexto relevante. El objeto RecognitionResult es entonces construido con base en este valor.



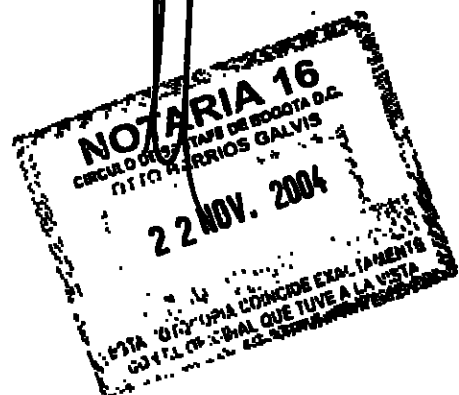
La clase Context Node puede incluir adicionalmente una propiedad titulada "ContextLink[] Links {get;}" que suministra una matriz de objetos ContextLink. Un objeto ContextLink describe una relación alternativa entre nodos de contexto. Aunque los nodos de contexto típicamente tienen una relación de padre a hijo con otros nodos de contexto, un ContextLink permite una relación alternativa entre nodos de contexto. Por ejemplo, un ContextLink puede permitir una conexión entre dos nodos de contexto, el anclaje de uno de los nodos de contexto a otro nodo de contexto, la contención de un nodo de contexto por otro nodo de contexto, o un tipo deseado de enlace definido por la aplicación de software. Los ContextLinks pueden ser agregados a esta matriz por medio de llamar el método AddLink. Similarmente, los ContextLinks pueden ser removidos de esta matriz por medio de llamar el método DeleteLink method.

Todavía aun, esta clase puede incluir las propiedades "IsContainer {get;}" y "IsInkLeaf {get;}" La propiedad IsContainer {get;} tiene el valor de "verdadero" si el nodo de contexto relevante no es un nodo de contexto de hoja (esto es, si el nodo de contexto



relevante contiene nodos de contexto hijos y es por lo tanto considerado un nodo de contexto de contenedor), y tiene el valor de "falso" de otra manera. La propiedad `IsInkLeaf {get;}` tiene el valor de "verdadero" si el nodo de contexto actual no es un nodo de contexto contenedor, y tiene el valor de "falso" de otra forma. Esto es, si el nodo de contexto actual no contiene ningunos nodos de contexto hijos, es considerado un nodo de contexto de hoja. En ciertas modalidades, un nodo de contexto `InkLeaf` es esperado que contenga referencias para golpear datos, mientras que los nodos de contexto de contenedor no tiene esta restricción. Los nodos de contexto de contenedor pueden o no referenciar golpes de datos o datos golpeados, como designado por la aplicación de software.

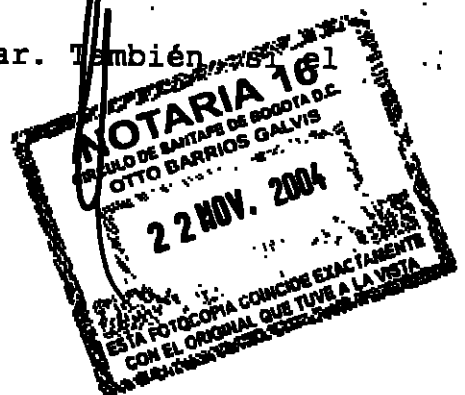
La clase de Nodo de Contexto puede también contener la propiedad `"Rect RotatedBoundingBox {get; set;}"`. El valor de esta propiedad es calculado por un diseño y proceso de análisis de clasificación. Si el dato de tinta asociado con el nodo de contexto relevante está escrito a un ángulo, entonces los límites para el nodo de contexto serán alineados horizontalmente. El valor de la propiedad `RotatedBoundingBox property`, sin embargo, estará alineada al ángulo en el cual el dato de tinta asociado con el



930

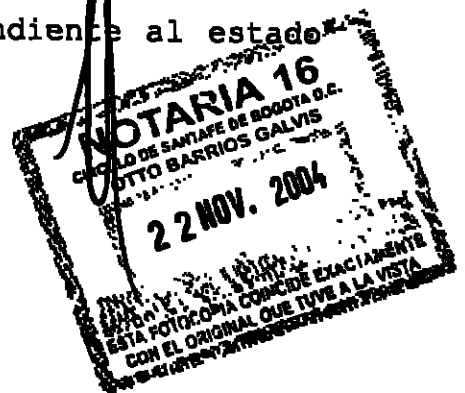
nodo de contexto relevante fue escrito. Adicionalmente, esta clase puede también incluir la propiedad "ReClassifiable {get;}," que informa al InkAnalyzer es permitido modificar los valores del nodo de contexto relevante.

En adición a estas propiedades, la clase de Nodo de Contexto puede también incluir diversos métodos. Por ejemplo, esta clase puede incluir un método titulado "ContextNode.CreateSubNode(ContextNodeType type)." Este método permite la creación de un nodo de contexto hijo de un tipo particular. En una modalidad, este método puede solamente permitir tipos de hijo válidos del nodo de contexto relevante que va hacer creado, impidiendo así estructuras de datos mal formadas que sean creadas. Por ejemplo, este método puede solamente permitir un nodo de contexto de línea para crear los nodos de contexto hijo InkWord y TextWord. Esta clase puede también contener un método titulado "void DeleteSubNode(ContextNode node)," el cual elimina los nodos de contexto hijo referenciados del objeto de análisis de contexto relevante. Sin embargo, en ciertas modalidades si el nodo de contexto referenciado todavía contiene golpes o nodo de contexto hijo, entonces este método debería fallar. También,



nodo de contexto referencia no es un hijo directo del nodo de contexto relevante, entonces este método debería fallar. Si una aplicación de software implementa su propio objeto de contexto de análisis y a su vez emplea este método, este no elimina los nodos de contexto no vacíos o nodos de contexto que no son hijos directos del nodo de contexto relevante para impedir estructuras de datos mal formadas en el objeto de contexto de análisis.

Adicionalmente, esta clase puede incluir el método "ContextNode[] HitTestSubNodes(AnalysisRegion region)," que regresa una matriz de nodo de contexto que están localizados en la región especificada. Sin embargo, solamente los nodos hijos inmediatos de este elemento son regresados, no todos los descendientes. La región está definida por el objeto AnalysisRegion, la cual puede ser una colección de uno o más rectángulos. En modalidades particulares, si cualquier parte de una localización de nodo de contexto intercepta la región especificada, entonces ese nodo de contexto será regresado a la matriz. Este método es empleado para, por ejemplo, crear el objeto de contexto de análisis independiente de documento y par a reconciliar los resultados del análisis con el objeto de contexto de análisis correspondiente al estado.



930
932

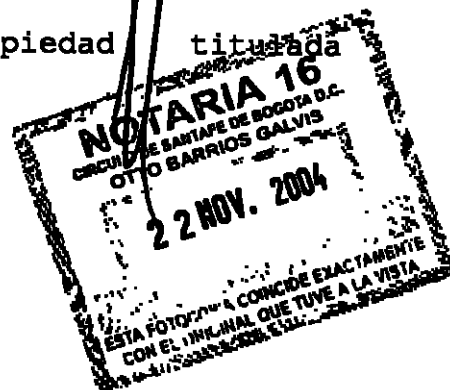
actual del documento. Así, este método es frecuentemente llamado, y debería ser optimizado par un acceso rápido repetido por el objeto InkAnalyzer.

La clase de Nodo de Contexto puede también contener un método titulado "MoveStroke(Stroke stroke, ContextNode destination)." Este método mueve la asociación de un golpe de un nodo de contexto de hoja a otro. En ciertas modalidades, este método es usado solamente entre nodos de contexto de hoja. Este puede incluir un método titulado "MoveSubNodeToPosition(int OldIndex, int NewIndex)," el cual reordena el nodo de contexto relevante con respecto a sus nodos de contexto hijos. Por ejemplo, si el documento tiene tres palabras en una línea, por ejemplo, palabra 1, palabra 2 y palabra 3, entonces su orden es implicado por la matriz de sub-nodos regresados del nodo de contexto padre. Este método permite que su orden sea cambiado de modo que, en relación al nodo de contexto padre relevante, la palabra 1 es especificada como la última palabra en la línea moviendo el nodo de contexto para la palabra 1 desde la posición 1 a la posición tres.



Adicionalmente todavía, esta clase puede incluir el método titulado "AddLink(ContextLink link)," que agrega un nuevo objeto ContextLink al nodo de contexto actual. En modalidades particulares, el objeto ContextLink debería contener una referencia al nodo de contexto relevante para poder hacer que el ContextLink sea agregado exitosamente a la matriz de los ContextLinks asociados al nodo de contexto relevante. Este puede también contener el método titulado "DeleteLink(ContextLink link)." Este método elimina o remueve el objeto ContextLink especificado de la matriz de ContextLinks para el nodo de contexto relevante. En una modalidad, este método llama siempre completamente de manera exitosa, aun si el ContextLink no existe en la matriz de ContextLinks asociados con el nodo de contexto relevante.

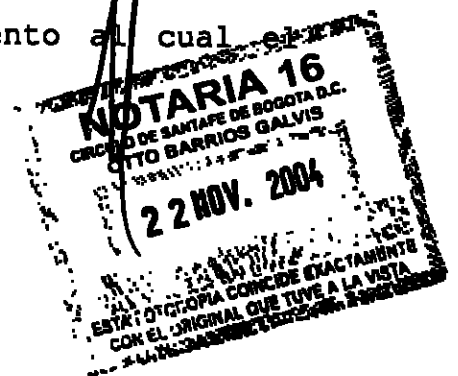
El Ink Analysis API puede también incluir una clase de Analysis Hint. Tal como con muchas de las clases previamente descrita, la clase Analysis Hint puede incluir un constructor, titulado "AnalysisHint()," el cual inicializa un objeto Analysis Hint para un estado vacío. Esta clase puede también incluir un número de propiedades, incluyendo una propiedad titulada



934

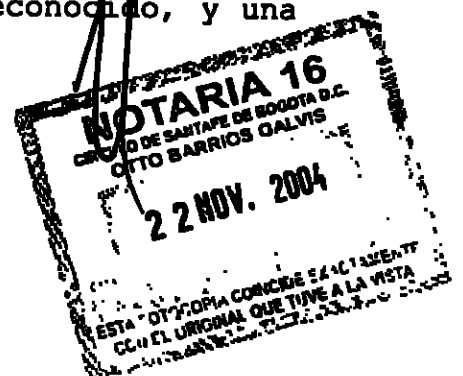
"AnalysisRegion Location {get;}." Esta propiedad especifica la localización en el documento (como un AnalysisRegion) al cual el AnalysisHint es aplicable. Por ejemplo, si el documento es una nota libre de forma con una sección de título en la parte superior de la página, entonces la aplicación podría establecer un AnalysisHint para la región de título para especificar que una línea horizontal de tinta se espera en esa región. Este Analysis Hint ayudará a incrementar la precisión de un proceso de análisis.

Esta clase también puede incluir una propiedad titulada "string Factoid {get; set;}," que especifica un "factoide" particular que es usado para la localización en el documento al cual el AnalysisHint es aplicable. Los Factoides suministran pistas a un proceso de reconocimiento en cuanto a un uso esperado de los datos de tinta (por ejemplo, un texto regular, dígitos, códigos postales, nombres de archivo y URLs de red). Esta clase puede incluir también las propiedades tituladas "RecognizerGuide Guide {get; set;}" y "OverrideLanguageId {get; set;}." La propiedad RecognizerGuide Guide {get; set;} especifica la guía de escritura que debe ser aplicada a la localización en el documento al cual se aplica.



`AnalysisHint` es aplicable. Las guías de escritura pueden, por ejemplo, ayudar a mejorar la precisión de un proceso de análisis de reconocedor por medio de especificar al usuario e informar el proceso de análisis del reconocedor en donde el usuario escribirá en líneas o caracteres. La propiedad `OverrideLanguageId {get; set;}` especifica un `Language Hint` para el documento al cual el `AnalysisHint` es aplicable. El establecimiento de un `Language Hint` causa que el objeto `InkAnalyzer` use el lenguaje designado en lugar del lenguaje especificado en el nodo de contexto.

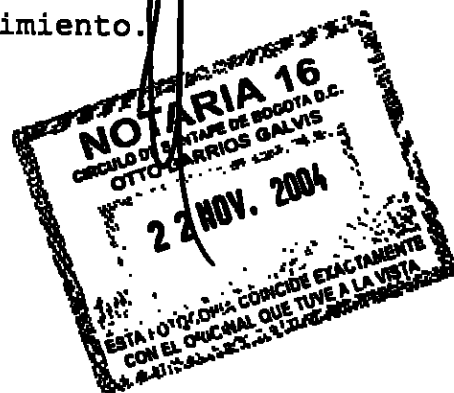
Esta clase también puede incluir una propiedad titulada `"PrefixText {get; set;}"` que especifica el texto que es escrito antes de una línea de tinta que va a ser reconocido. Todavía adicionalmente esta clase puede incluir una propiedad titulada `"RecognitionModes RecognitionFlags {get; set;}"` que especifica un tipo particular de modos en que un proceso de reconocimiento debe respetar en una localización en el documento para el cual el `AnalysisHint` es aplicable. Adicionalmente, esta clase puede incluir una propiedad titulada `"SuffixText {get; set;}"` que especifica que el texto que es escrito después de una línea de tinta va a ser reconocido, y una



936

propiedad titulada "WordList WordList {get; set;}," que especifica un conjunto particular de palabras que debería ser usados por un proceso de análisis de reconocimiento. Las listas de palabras pueden ser usadas cuando los resultados del reconocimiento esperados son conocidos antes de que el usuario halla escrito realmente los datos de entrada, tal como una lista de términos médicos que son esperados que se escriban al interior de una fórmula médica.

Todavía adicionalmente, esta clase puede incluir una propiedad titulada "WordMode {get; set;}. Si este valor "verdadero", entonces el proceso de análisis se empujará así mismo para regresar una palabra única en toda la región de análisis. Esto puede también incluir una propiedad titulada "Coerce {get; set;}," que, si es "verdadero", forzará al proceso de análisis para confinar su resultado a cualquier factotipo valor de lista de palabra establecido en la pista relevante. Esta clase también puede tener incluida una propiedad titulada "AllowPartialDictionaryTerms {get; set;}. Si este valor de propiedad es "verdadero" entonces el proceso de análisis de reconocimiento permitirá regresar palabras parciales de su diccionario de reconocimiento.



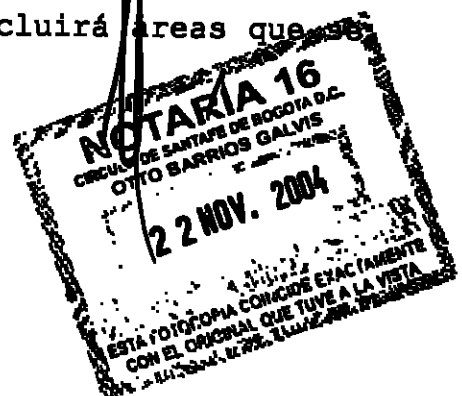
937

En modalidades particulares, el Ink Analysis API puede incluir adicionalmente una clase de Región de Análisis. Esta clase puede incluir, por ejemplo, múltiples constructores para construir un objeto de AnalysisRegion. Por ejemplo, este puede contener un primer constructor para construir un objeto AnalysisRegion que tiene cualquier región, un segundo constructor para construir un objeto AnalysisRegion con base en los parámetros para un rectángulo bidimensional, y un tercer constructor para construir un objeto AnalysisRegion con base en coordenadas de cuatro espacios. El constructor por defecto puede, por ejemplo, crear una región vacía. Esta clase puede también incluir un número de propiedades. Por ejemplo, esta clase puede incluir una propiedad titulada "Rectangle Bounds {get;}," la cual recupera los límites del rectángulo para el AnalysisRegion, una propiedad titulada "IsEmpty {get;}," que indica si el objeto AnalysisRegion tiene un interior vacío, y una propiedad titulada "IsInfinite {get;}," que indica si el AnalysisRegion relevante está establecido en infinito o no.



Esta clase puede también incluir un número de métodos tal como un método titulado "AnalysisRegion Clone()," el cual clona el objeto AnalysisRegion relevante. Esta clase puede también incluir un método titulado "Equals(AnalysisRegion otherRegion)," el cual verifica si el objeto de AnalysisRegion especificado (referido como el otherRegion) es idéntico al objeto AnalysisRegion relevante. Este método regresa un valor de "verdadero" si el interior del objeto Analysis Region especificado es idéntico al interior el objeto Analysis Region relevante, y de otra manera regresa un valor de "falso".

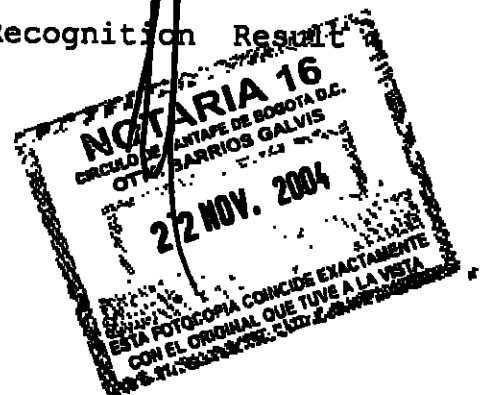
Esta clase puede también incluir adicionalmente un método "Intersect(AnalysisRegion regionToIntersect)," el cual barre hacia abajo el objeto AnalysisRegion relevante a la región de análisis especificada. Así, el objeto AnalysisRegion resultante incluirá solamente áreas que sobreponen o interceptan la región de análisis especificada. Esta clase puede también incluir el método titulado "Intersect(Rectangle rectangle)," el cual barre hacia abajo el objeto AnalysisRegion relevante al rectángulo especificado. Nuevamente, el objeto AnalysisRegion resultante solamente incluirá áreas que



sobreponen o interceptan en el rectángulo especificado. Esto puede incluir el método titulado como "MakeEmpty()," el cual inicializa el objeto AnalysisRegion relevante a un interior vacío, y el método titulado "MakeInfinite()," el cual establece el área ocupada por el AnalysisRegion para que sea infinito.

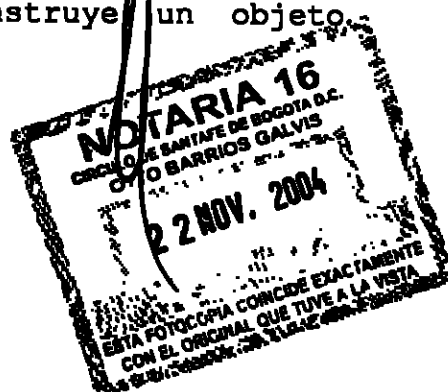
Esto puede incluir adicionalmente varios métodos para unificar o separar de manera diferente áreas definidas, tal como el método titulado "Union(AnalysisRegion regionToUnion)," el cual especifica un objeto AnalysisRegion para unir o agregar al objeto AnalysisRegion relevante, y el método titulado "Union(Rectangle rectangle)," el cual une un rectángulo especificado al objeto Analysis Region relevante. Con este método, el rectángulo puede ser especificado en términos de coordenadas espaciales para el objeto AnalysisRegion relevante. Por supuesto, esta clase puede también incluir numerosos otros métodos para combinar áreas, o extraer un área de otra o base en la definición deseada en cualquier definición deseada para las áreas.

El Ink Analysis API puede también tener una clase Recognition Result. Tal como con muchas de las clases discutidas anteriormente, la clase Recognition Result



quo

puede incluir uno o más constructores. Por ejemplo, esta clase puede incluir un constructor titulado "RecognitionResult(Stream lattice)", que construye un objeto RecognitionResults de una red de reconocimiento es una forma serializada de resultados desde un proceso de reconocimiento. Este método puede, por ejemplo, especificar una red de reconocimiento como una matriz de bytes para ser usada por la construcción del objeto RecognitionResult relevante. Esta también puede incluir un constructor titulado "RecognitionResult(ContextNode node)," el cual construye un objeto RecognitionResults de un nodo de contexto dado. Este también puede incluir un constructor titulado "RecognitionResult(string Text, int StrokeCount)," el cual construye un objeto RecognitionResults de un valor de texto especificado, el cual a su vez está asociado con un número especificado de golpes y puede ser usado para corregir si el proceso de reconocimiento o llegó con un valor de reconocimiento alternativo correspondiente al dato de tinta manuscrita actual. Todavía adicionalmente, esta clase puede incluir un constructor titulado "RecognitionResult(RightRecognitionResult leftRecognitionResult, RecognitionResult rightRecognitionResult)," el cual construye un objeto



441

RecognitionResults por medio de fundir o unir dos objetos Recognition Results existentes juntos.

La clase Recognition Result puede incluir una o más propiedades, tal como una propiedad titulada "StrokeCollection StrokeCollection {get;}," la cual suministra una matriz de índices de golpes que representan una colección de golpes que están contenidos en un objeto de tinta único, y una propiedad titulada "RecognitionAlternate TopAlternate {get;}," la cual suministra la mejor alternativa del resultado de reconocimiento. Esta clase puede también incluir la propiedad titulada "RecognitionConfidence RecognitionConfidence {get;}," la cual suministra el nivel de confianza (por ejemplo, fuerte, intermedia, o pobre) de la selección alternativa superior para los resultados actuales de un proceso de análisis de reconocimiento, y una propiedad titulada "string TopString {get;}," que regresa la mejor cadena de resultado de los resultados del análisis de un proceso de análisis de reconocimiento.

La clase Recognition Results puede también incluir un número de métodos, tal como un método titulado "public RecognitionAlternateCollection



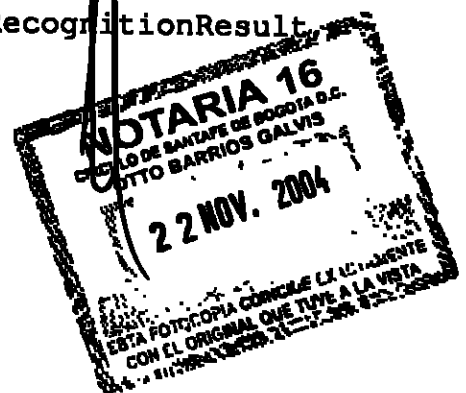
340

245

942

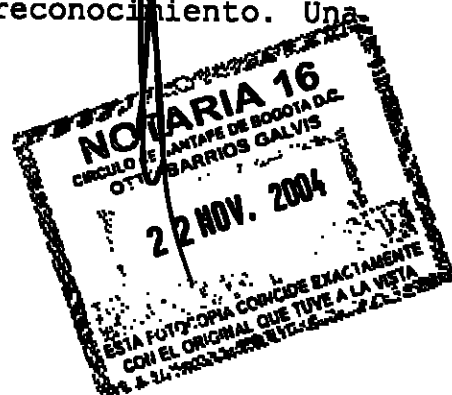
`GetAlternateCollectionFromSelection` (`selectionStart`, `selectionLength`, `maximumAlternates`), "que especifica una colección de alternativas de una selección dentro de la cadena de resultado mejor de los resultados de análisis de un proceso de análisis de reconocimiento, en donde cada alternativa corresponde a solamente un segmento de tinta. Los parámetros de entrada para este método pueden incluir, por ejemplo, un valor que especifica el inicio de la selección de texto desde la cual la colección de alternativas es regresada, un valor que especifica la longitud de la selección del texto desde el cual la colección de alternativas es regresada, y un valor que especifica el número máximo de alternativas que se deben regresar. Este método puede entonces regresar o retornar la colección de alternativas `RecognitionAlternateCollection` de una selección dentro de la mejor cadena resultante del resultado de reconocimiento.

La clase `Recognition Results` puede incluir adicionalmente un método titulado "`RecognitionResult Merge(RecognitionResult left, string separator, RecognitionResult right)`." Este método puede ser usado para crear un nuevo objeto `RecognitionResult`.



qu3

introduciendo una cadena única al comienzo o al final de un objeto `RecognitionResult` existente, o concatenando una cadena única entre dos objetos `RecognitionResult` existentes. Esta clase puede también incluir un método titulado `"ModifyTopAlternate(RecognitionAlternate, alternate),"` que especifica el resultado de reconocimiento hacer modificado con una alternativa conocida. Con algunas modalidades, por defecto la mejor cadena resultante de los resultados de un proceso de análisis de reconocimiento corresponde a la alternativa superior. Sin embargo, este método puede ser usado para especificar que alternativas diferentes de la alternativa superior son usadas en los resultados del proceso de análisis de reconocimiento. Si la alternativa nueva superior resulta en una segmentación diferente que la anterior, el método `ModifyTopAlternate` actualizará automáticamente los nodos de contexto para reflejar los cambios. Para recuperar las alternativas que pueden ser usadas para modificar el resultado de reconocimiento, este método se llama el método `GetAlternatesFromSelection`. Esta clase también puede tener un método titulado `"Stream Save(),"` que persistentemente mantiene el objeto `RecognitionResults` relevante en la forma de una red de reconocimiento. Una



qu4

red de reconocimiento es un formato serializado usado para expresar los resultados de un proceso de reconocimiento.

El Ink Analysis API Puede también tener un tipo enumerado de Analysis Options. Este tipo puede contener uno o más campos que especifican como los datos de tinta serán analizados por un proceso de análisis, tal un campo titulado "const AnalysisOptions Default," que permite todas las opciones disponibles para el proceso de análisis. Este campo puede, por ejemplo, permitir reconocimiento de texto, el uso de tablas, el uso de listas, el uso de anotaciones, el uso de conectores y contenedores, y el uso de resultados intermedios. Este tipo también puede incluir un campo titulado "const AnalysisOptions EnableAnnotations," que permite y deshabilita la detección de anotaciones, un campo titulado "const AnalysisOptions EnableConnectorsAndContainers," el cual y permite y deshabilita la detección de conectores y contenedores, y un campo titulado "const AnalysisOptions EnableIntermediateResults," que permite y deshabilita el regreso de los resultados de análisis a la aplicación de software entre el uso de diferentes procesos de análisis.



945

secuenciales (por ejemplo, entre un proceso de análisis y un proceso de reconocimiento subsecuente). Este tipo puede también tener un campo titulado "const AnalysisOptions EnableLists," que permite y deshabilita la detección de listas y un campo titulado "const AnalysisOptions EnableTables," que permite y deshabilita la detección de tablas. Este tipo enumerado puede incluir adicionalmente un campo titulado "const AnalysisOptions EnableTextRecognition," que permite y deshabilita un proceso de análisis de reconocimiento de texto. Sin embargo, si procesos de análisis adicionales están disponibles (o diferentes versiones del mismo proceso de análisis), entonces este tipo puede incluir AnalysisOptions adicionales de acuerdo con lo anterior.

Todavía adicionalmente, el Ink Analysis API puede incluir una clase AnalysisResultsEventArgs. Esta clase puede tener un constructor titulado "public AnalysisResultsEventArgs()," el cual crea una estructura de datos que contiene los resultados de análisis y es regresado a la aplicación de software cuando el evento AnalysisResults es elevado. Esta clase puede también incluir una propiedad titulada "InkAnalyzer InkAnalyzer

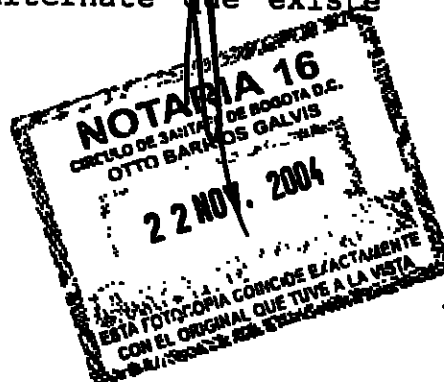


455
946

{get;}," la cual identifica el objeto InkAnalyzer que llevo a cabo el proceso de análisis.

El API puede también tener una clase Line, la cual puede ser útil con algunos tipos de sistemas operativos que reconocen el uso de un objeto "Line" que representa una línea geométrica. Esta clase puede incluir un constructor tal como un constructor titulado "public Line(Point beginPoint, Point endPoint)," el cual crea un objeto Line. Esta clase puede también incluir varias propiedades, tal como una propiedad titulada "public Point BeginPoint {get; set;}," el cual representa el punto de inicio del objeto línea y una propiedad titulada "public Point EndPoint {get; set;}," el cual representa el punto final del objeto línea.

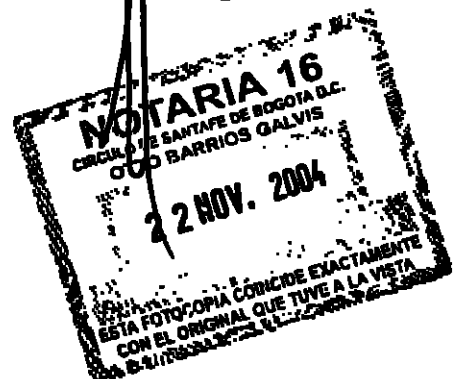
En adición a estas clases, el Ink Analysis API puede también contener una clase Recognition Alternate. Esta clase puede incluir elementos que representan las posibles cases de palabras para segmentos de tinta que son comparados por un diccionario del reconocedor. Por ejemplo, esta clase puede incluir una propiedad titulada "Line Ascender {get;}," la cual suministra la línea ascendente de un objeto RecognitionAlternate que existe



947

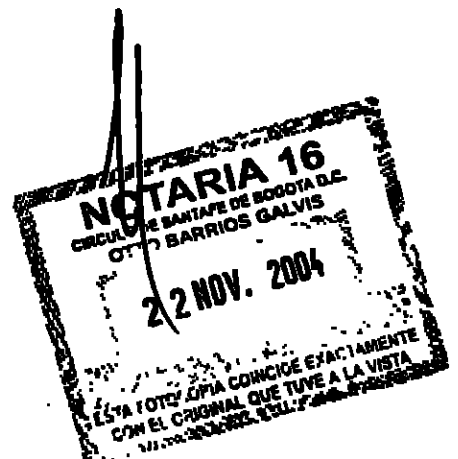
sobre una línea única (con una línea siendo representada como dos puntos), una propiedad titulada "public Line Baseline {get;}," el cual suministra la línea base de un objeto RecognitionAlternate que existe sobre una línea única, y una propiedad titulada "Line Descender {get;}," que suministra la línea descendente de un objeto RecognitionAlternate que existe sobre una línea única. Esta clase puede también incluir una propiedad titulada "RecognitionResult Extract {get;}," la cual suministra un objeto RecognitionResults para el objeto RecognitionAlternate actual. Esta propiedad puede ser usada, por ejemplo, para extraer el objeto RecognitionResult para una palabra desde el objeto RecognitionResult para una línea que contiene esa palabra.

Esto puede incluir también la propiedad entitulada "Line Midline {get;}," la cual suministra la línea media para un objeto RecognitionAlternate que existe sobre una línea única, una propiedad titulada "StrokeCollection Strokes {get;}," la cual suministra la colección de golpes que están contenidos en un objeto tinta (que este suministra un, StrokeCollection que representa los golpes que están asociados con el RecognitionResult), y una



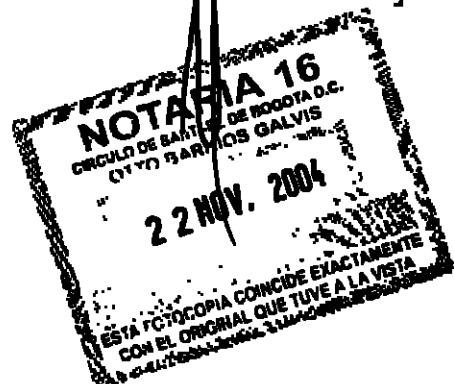
948

propiedad titulada "StrokeCollection[] StrokesArray {get;}" la cual suministra una colección de golpes que están contenidos en uno o más objetos de tinta, que representan los golpes que están asociados con el RecognitionResult. Esta clase también puede incluir una propiedad titulada "RecognitionConfidence RecognitionConfidence {get;}" la cual provee el nivel de confianza (por ejemplo, fuerte, intermedia, o pobre) de un proceso de análisis de reconocimiento a determinado en el reconocimiento de un objeto RecognitionAlternate o de un gesto. Para nodos no-lineales, el RecognitionConfidence más bajo del hijo de los nodos de contexto relevantes será regresado. Este puede contener también la propiedad titulada "string RecognizedString {get;}" la cual especifica la cadena resultante de la alternativa. Así, para cualquier nodo de contexto anterior a un nodo de contexto de palabra, la cadena que resulta esta concatenada junta por este método. Por ejemplo, un nodo de línea contendrá una cadena resultante que a su vez contiene los resultados de todos los hijos o nodos de palabra. Un nodo de párrafo entonces contendrá una cadena de resultados que contiene los resultados de todos sus hijos o nodos de línea.



449

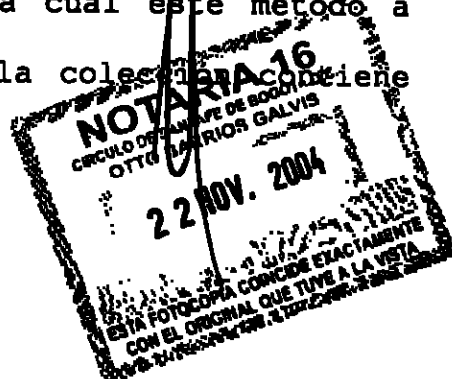
La clase Recognition Alternate también puede contener uno o más métodos incluyendo, por ejemplo, un método titulado "StrokeCollection[] GetStrokesArrayFromTextRange(int selectionstart, int selectionlength)," el cual especifica un StrokeCollection para cada objeto de tinta que corresponde al rango de texto conocido. Esta clase también puede contener un método titulado "StrokeCollection[] GetStrokesFromStrokesArrayRanges(StrokeCollection[] strokesArray)," el cual especifica la colección más pequeña de golpes que contiene una colección de golpes de entrada conocida y para la cual el reconocedor puede suministrar alternativas. Más particularmente, los golpes son regresados por una matriz de objetos de tinta que contiene cada uno una matriz de golpes indexados para la colección. La colección de golpes de tinta regresada por este método puede casar con la colección de entradas, o puede ser más grande si la colección de entradas casa solamente con parte del resultado de reconocimiento más pequeño que incluye todos los golpes de entrada. Esta clase puede incluir adicionalmente un método titulado GetStrokesFromStrokesRanges(StrokeCollection strokes)," que especifica la colección más pequeña de golpes que contiene una colección de golpes de entrada conocida y



450

para la cual el reconocedor puede suministrar alternativas, y un método titulado "StrokeCollection GetStrokesFromTextRange(int selectionstart, int selectionlength)," que especifica el StrokeCollection que corresponde al rango de texto conocido.

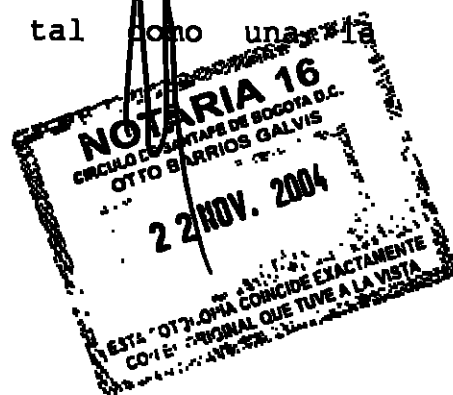
Esta clase puede incluir adicionalmente un método titulado "void GetTextRangeFromStrokes(ref int selectionstart, ref int selectionend, StrokeCollection strokes)," el cual especifica el rango más pequeño del texto reconocido para el cual el reconocedor puede retornar una alternativa que contiene un conjunto conocido de golpes, y un método titulado "void GetTextRangeFromStrokesArray(ref int selectionstart, ref int selectionend, StrokeCollection[] strokesarray)," el cual especifica el rango más pequeño del texto reconocido para el cual el reconocedor puede retornar una alternativa que contiene un conjunto conocido de golpes. Este también puede tener un método titulado "RecognitionAlternateCollection SplitWithConstantPropertyValue(GUID propertyType)," el cual regresa una colección de alternativas, que son una división de la alternativa sobre la cual este método a sido llamado. Cada alternativa en la colección tiene



951

segmentos de reconocimiento adyacentes que tiene el mismo valor de propiedad para la propiedad pasada al método. Por ejemplo, este método puede ser usado para obtener alternativas que dividen una alternativa original por nivel de límites de confianza (fuerte, intermedio, o pobre) en el resultado de reconocimiento, los límites de línea, o límites de segmento. Este puede incluir adicionalmente un método titulado "byte[] GetPropertyValue(GUID propertyType)," que especifica el valor de una propiedad conocida de la alternativa, tal como la confianza del reconocedor en la alternativa. No todos los procesos de análisis de reconocimiento suministrarán un valor para todos los tipos de propiedades, sin embargo. Así, este método suministra los datos para los tipos soportados por el proceso de análisis de reconocimiento relevante.

El Ink Analysis API puede también incluir una clase de colección de Recognition Alternate. Como muchas de las clases discutidas anteriormente, esta clase puede incluir un constructor, titulado "RecognitionAlternateCollection()," para crear un objeto RecognitionAlternateCollection. Esta clase puede también incluir un número de propiedades, tal como una

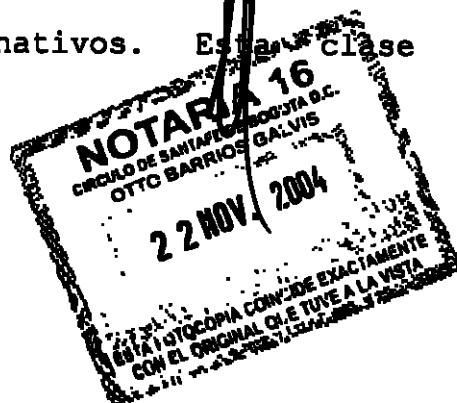


350

200
952

propiedad titulada "Count {get;}," que suministra el número de objetos o colecciones contenidas en una colección de valores de reconocimiento alternativos, una propiedad titulada "IsSynchronized {get;}," que suministra un valor que indica si el acceso a la colección de valores de reconocimiento alternativos esta sincronizado con la aplicación del software (es decir, "hilo seguro"), y una la propiedad titulada "SyncRoot {get;}," que suministra el objeto que puede ser usado para sincronizar el acceso a la colección de valores de reconocimiento alternativos.

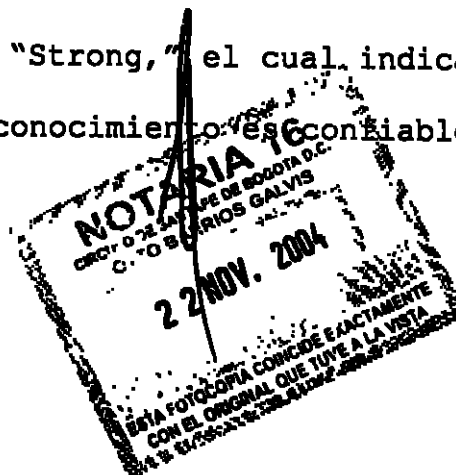
Esta clase puede también contener uno o más métodos, tal como un método titulado "virtual void CopyTo(Array array, int index)," el cual copia todos los elementos de la colección actual de valores de reconocimiento alternativos a la matriz de una dimensión especificada, iniciando en el índice de matriz de destino especificado, y un método titulado "IEnumerator IEnumerable.GetEnumerator()," el cual es una implementación estándar de IEnumerable permite que los llamadores lo usen para cada construcción para enumerar a través de cada RecognitionAlternate en la colección de valores de reconocimiento alternativos. Esta clase



453

también puede incluir un método titulado "RecognitionAlternateCollectionEnumerator GetEnumerator()," el cual retorna un RecognitionAlternateCollectionEnumerator que contiene todos los objetos dentro de la colección de los valores alternativos de reconocimiento. Este método puede ser usado, por ejemplo, para recuperar cada objeto en la colección de valores alternativos de reconocimiento.

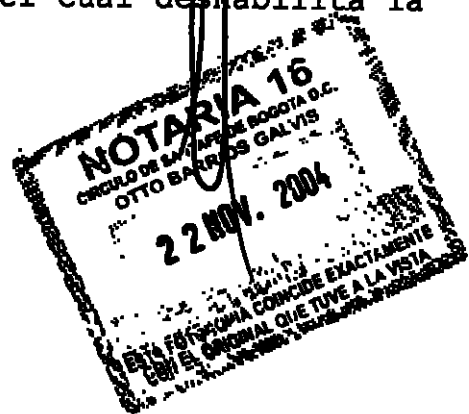
El Ink Analysis API puede adicionalmente incluir una enumeración Recognition Confidence y una enumeración Recognition Mode enumeration, cada una de las cuales puede contener uno o más campos relacionados con un proceso de análisis de reconocimiento. Por ejemplo, la clase Recognition Confidence puede contener múltiples campos, tal como un campo titulado "Intermediate," que indica que el proceso d análisis de reconocimiento es confiable en que el resultado correcto está en la lista de valores de reconocimiento alternativo suministrada, un campo titulado "Poor," que indica que el análisis de reconocimiento no es muy confiable en que el resultado esté en la lista de valores de reconocimiento alternativo, y un campo titulado "Strong," el cual indica que el proceso de análisis de reconocimiento es confiable



954

en que la mejor alternativa esta en los valores de reconocimiento alternativos es la correcta.

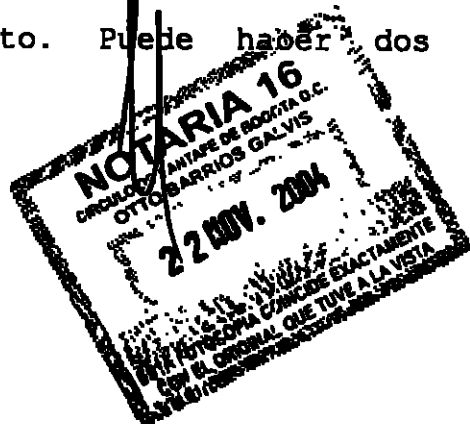
Similarmente, la clase Recognition Mode puede incluir campos que especifican como un proceso de análisis de reconocimiento interpreta los datos de tinta electrónica y por lo tanto determina una cadena de resultado de reconocimiento. Por ejemplo, esta clase puede incluir un campo titulado "Coerce," el cual especifica que el proceso de análisis de reconocimiento contiene un resultado de reconocimiento con base en un factor de que fue especificado para el contexto, y un campo titulado "Line," que especifica que el proceso de análisis de reconocimiento trata el dato de tinta electrónica como una línea única. Esta clase también puede incluir un campo titulado "None," el cual especifica que el proceso de análisis de reconocimiento aplica a ningún modo de reconocimiento, y un campo titulado "Segment," que especifica que el proceso de análisis de reconocimiento trata el dato de tinta electrónica como formar una palabra única o un carácter único. Todavía adicionalmente esta clase puede incluir un campo titulado "TopInkBreaksOnly," el cual deshabilita la segmentación múltiple.



955

Todavía adicionalmente, el Ink Analysis API puede incluir una clase Context Link class, la cual define como dos nodos de contexto pueden ser enlazados juntos. El objeto ContextLink en si mismo representa cuales de los dos nodos de contexto están ligados o enlazados, la dirección del enlace, y el tipo de enlace. Esta clase puede incluir una propiedad titulada "ContextNode SourceNode{get;}," la cual especifica el nodo de contexto fuente que está siendo enlazado desde otro nodo de contexto, una propiedad titulada "ContextLinkType LinkType{get;}," que especifica el tipo de relación de enlace que existe entre los nodos de contexto fuente y destino, y una propiedad titulada "CustomLinkType{get;}," que especifica que un enlace adaptado está siendo usado. Esta situación ocurriría cuando una aplicación decide usar el sistema de enlace del Ink Analyzer API para representar la aplicación de enlaces específicos más allá de lo que el API puede reconocer.

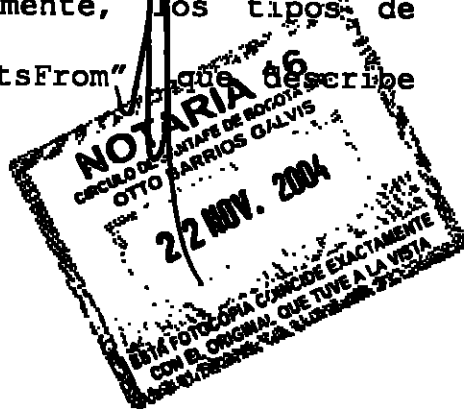
Esta clase también puede incluir una propiedad titulada ContextNode DestinationNode {get;}," la cual especifica el nodo de contexto destino que está siendo ligado de otro nodo de contexto. Puede haber dos



956

constructores disponibles para esta clase, los cuales crean una relación entre los nodos de contexto fuente y destino existentes.

Esta clase también puede incluir una enumeración titulada "ContextLinkType enum," la cual define el tipo de relación compartida por dos nodos de contexto. Estos tipos de enlace diversos pueden incluir, por ejemplo, un tipo "AnchorsTo", el cual describe que un nodo está anclado al otro nodo. Ambos nodos pueden usar las propiedades SourceNode o DestinationNode con base en la situación de análisis. En los tipos de enlace también pueden incluir el tipo "Contains," el cual describe que el un nodo contiene el otro nodo. Con esta relación, el nodo contenedor podría ser referenciado como el SourceNode, mientras que el nodo contenido podría ser referenciado como el DestinationNode. Los tipos de enlace pueden incluir adicionalmente un tipo "PointsTo", el cual describe que un nodo está apuntando a otro nodo. Para esta relación, el nodo que hace el apuntamiento podría ser referenciado como el SourceNode, mientras que el nodo que está siendo apuntado podría ser referenciado como el DestinationNode. Todavía adicionalmente, los tipos de enlace pueden tener un tipo "PointsFrom" que describe



957

que un nodo está apuntando desde el otro nodo. En esta relación, el nodo que apunta hacia afuera del otro nodo podría ser referenciado como el `SourceNode`, mientras que el nodo que está siendo apuntado desde podría ser referenciado como el `DestinationNode`.

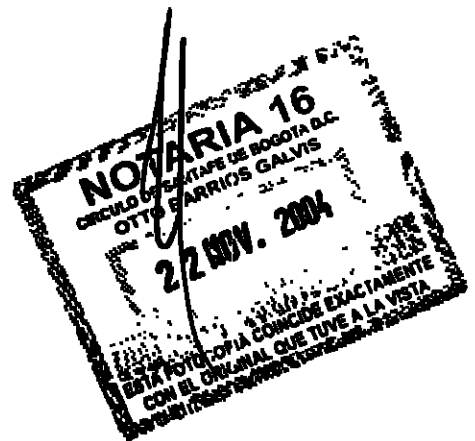
Los tipos de enlace pueden adicionalmente incluir un tipo `"SpansHorizontally"`, el cual describe que un nodo corre la longitud horizontalmente del otro nodo, y un tipo `"SpansVertically"`, el cual describe que un nodo corre la longitud verticalmente del otro lado. Para estos tipos, el nodo cubre (resolución, sub-línea, barra de margen) el otro nodo, usualmente escrito de último, que podría ser referenciado como el `SourceNode`, mientras que el nodo que está siendo barrido podría ser referenciado como el `DestinationNode`. Los tipos de enlace también pueden incluir un tipo `"Custom"`, el cual describe que un tipo de enlace adaptado a sido usado. Cuando este valor es usado, la propiedad `"CustomLinkType"` sobre el objeto `ContextLink` podría suministrar más detalles en cuanto al propósito de este enlace.

Application Model



958

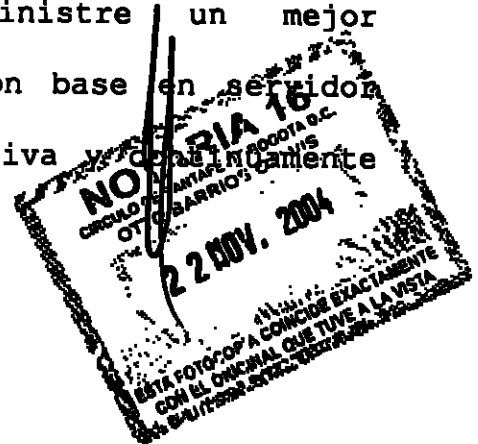
Un Cliente Windows integra características de la red con características de aplicaciones de escritorio tradicionales. EL Application Model suministra una estructura para aplicaciones seguras y simplifica el desarrollo, el despliegue y el mantenimiento de aplicaciones cliente. Esta estructura suministra una experiencia de usuario simple y consistente. Por ejemplo, aplicaciones locales pueden nivelar rasgos familiares de buscador sin importarle si la aplicación está hospedada en el buscador o si es una aplicación libre, aunque retiene los beneficios de ejecutarse sobre el cliente local. Esta estructura le permite a los usuarios nivelar su familiaridad con la red, incrementando así el nivel de comodidad del usuario y reduciendo el tiempo requerido para aprender a usar una nueva aplicación. La Application Model es parte del System.Windows namespace.



959

Las aplicaciones que utilizan la Application Model operan de una manera similar a las páginas Web. Cuando un usuario busca en una aplicación, o navega en una aplicación, la aplicación es instalada automáticamente sin requerir la confirmación del usuario de instalación, reiniciando el sistema de cliente, o teniendo riesgo de un mal funcionamiento de otras aplicaciones. En una modalidad, las aplicaciones se descargan progresivamente, suministrando así un nivel inicial de interactividad antes de que la aplicación esté completamente descargada. Actualizaciones de la aplicación se manejan automáticamente y en una manera que es transparente al usuario. Así, el usuario tiene siempre acceso a la última versión de la aplicación sin tener que llevar a cabo explícitamente una actualización de la aplicación.

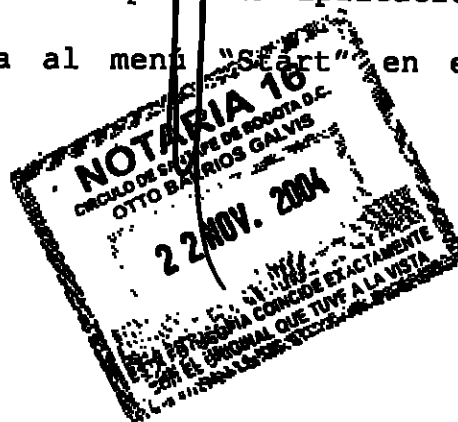
Las aplicaciones que usan el Application Model corren localmente sobre el sistema del cliente sin importar de si el sistema del cliente está en línea (es decir, acoplado activamente a la red) o fuera de línea (es decir, no acoplado activamente a la red). Esto permite que una aplicación suministre un mejor comportamiento que una aplicación con base en servidor que requiere una conexión de red activa y de alta velocidad.



960

intercambia datos con el servidor a través de la red. Después de que una aplicación es instalada sobre un sistema de cliente, la aplicación puede ser accedida desde un menú "Start" (similar a una aplicación de escritorio tradicional) o por medio de navegar a aplicación (similar a una aplicación de red). La Application Model contiene tres partes principales: una administración de ciclo de vida de la aplicación, una estructura de la aplicación y una estructura de navegación.

Dos tipos diferentes de aplicaciones están soportadas por el Application Model: una "aplicación en línea" y una "aplicación administrada". Las aplicaciones que utilizan la Application Model pueden ejecutarse en el buscador o en una ventana libre de nivel superior. Una "aplicación en línea" es una aplicación que se ejecuta desde servidor y está hospedada en un buscador. La aplicación puede estar escondida o guardada para acceso fuera de línea o la aplicación puede requerir ciertos recursos en línea para ejecutarse adecuadamente. Una "administración administrada" está disponible fuera de línea y es instalada en el cliente. El sistema operativo sirve a la aplicación administrada. Una entrada para la aplicación administrada puede ser agregada al menú "Start" en el

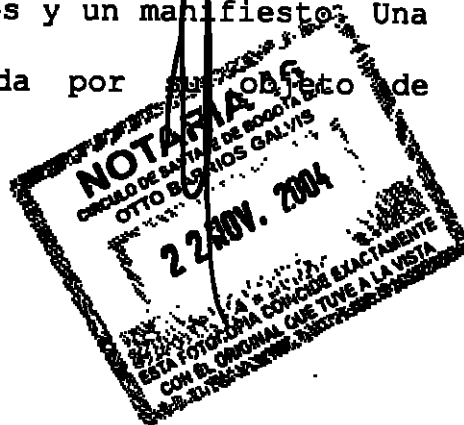


961

cliente. Las aplicaciones pueden ser descargadas progresivamente para permitir que el usuario inicie la interacción con la aplicación a medida que esta siendo descargada en vez de retrasar la interacción hasta que un proceso de instalación ha sido terminado.

Cuando las aplicaciones tienen una aplicación asociada manifiesta que describe las dependencias de la aplicación, tal como librerías adicionales recursos adicionales requeridos para ejecutar la aplicación. Un instalador usa el manifiesto de aplicación para controlar la descarga e instalación de la aplicación. Un "administrador confiable" es invocado como parte del proceso de instalación. El administrador confiable usa el manifiesto de aplicación para determinar que permisos son requeridos por la aplicación para ejecutarse. El manifiesto de aplicación también especifica la información de concha, tal como asociaciones de archivo y si se agrega una entrada al menú inicio lo mismo que el icono y texto para las entradas.

Las aplicaciones que utilizan la Application Model incluyen marcación, código, recursos y un manifiesto. Una aplicación es definida y limitada por

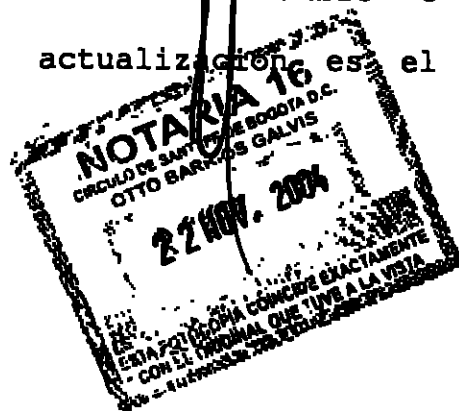


320

280
-
962

aplicación, el cual es un objeto global que persiste en memoria para el tiempo de vida de cada sección de la aplicación. El objeto de la aplicación tiene conocimiento de todos los recursos que pertenecen a la aplicación y suministra un límite entre si misma y otras aplicaciones o recursos externos. La estructura de la aplicación utiliza el objeto de aplicación para identificar, referenciar y comunicarse con la aplicación. El objeto de aplicación también es usado dentro de la aplicación para administrar ventanas y recursos, especifica el inicio y el cierre del comportamiento, maneja los parámetros de configuración, especifica estilos visuales para la aplicación, código compartido, estado y recursos a través de las navegaciones, y maneja los eventos de aplicación amplia.

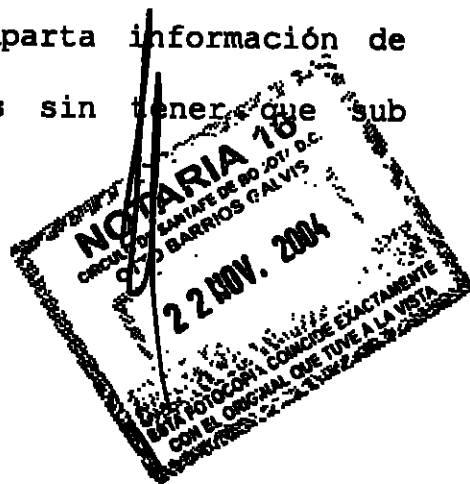
Una estructura de navegación soporta aplicaciones con base en navegación que nivelan la familiaridad del usuario con la navegación y actualizan las actividades sobre la red para suministrar una experiencia de usuario más familiar y consistente sobre el sistema del cliente, sin importarle si la aplicación está hospedada en un navegador del sistema o en una ventana libre o independiente de alto nivel. La actualización es el



963

proceso usado por la estructura de navegación para rastrear la historia de navegación. El actualizador permite a los usuarios volver a trazar sus etapas anteriores y hacia delante en una secuencia de navegación lineal. Si una experiencia de navegación es o esta hospedada en un navegador o en un buscador o en una ventana de navegación independiente, cada navegación es colocada en el actualizador y puede ser revisitada en una secuencia lineal por medio de usar los botones "forward" y "back" o por medio de invocar los métodos "go forward" y "go back". Cada ventana de navegación tiene asociado un actualizador.

Una clase NavigationApplication puede simplificar la tarea de crear aplicaciones con base en navegación por medio de suministrar propiedades y eventos relacionados con la navegación. La clase NavigationApplication incluye una propiedad de inicio que especifica la página o elemento en el cual el sistema navega cuando la aplicación es lanzada la primera vez. Esta clase también tiene una colección de propiedades que permite que un desarrollador de la aplicación comparta información de estado global a través de páginas sin tener que sub



964

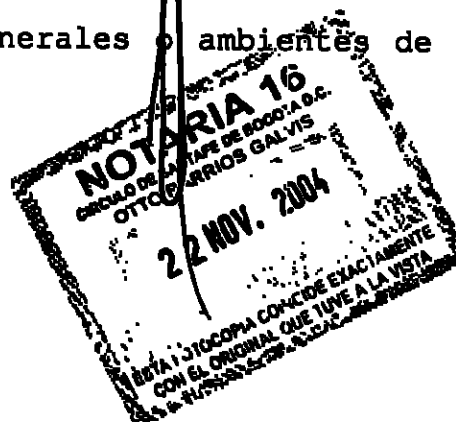
clasificar la aplicación, y soporta datos ligados a estas propiedades.

SISTEMA Y AMBIENTE DE CÓMPUTO ILUSTRATIVO.

La figura 4 ilustra un ejemplo de un ambiente de computo adecuado 400 dentro del cual la estructura de programación 132 puede ser implementada (sea completa o parcialmente). El ambiente de computo 400 puede ser utilizado en las arquitecturas del computador y red descritas aquí.

El ambiente de computo ilustrativo 400 es solamente un ejemplo de un ambiente de computo y no tiene el propósito de sugerir ninguna limitación en cuanto al alcance del uso o funcionalidad de las arquitecturas de computadora y red. Tampoco debería el ambiente de computo 400 ser interpretado como que tiene cualquier dependencia o requerimiento relacionado con una cualquiera o combinación de componentes ilustrados en el ambiente de computo ilustrativo 400.

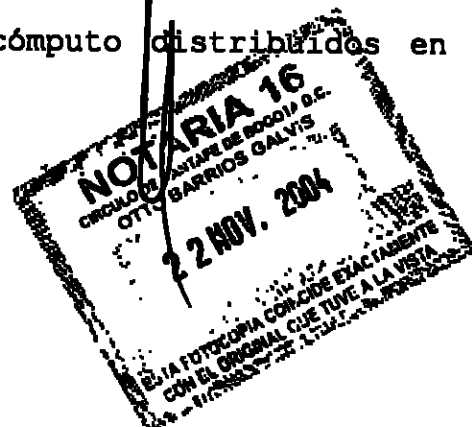
La estructura 132 puede ser implementada con numerosos propósitos diferentes generales o ambientes de



965

sistema de cómputo de propósito especial o configuraciones especiales. Ejemplos de sistemas de computo bien conocidos, ambientes, y/o configuraciones que pueden ser adecuados para el uso incluyen, pero no se limitan a, computadores personales, computadores servidores, sistemas multiprocesadores, PC de red, mini computadores, computadores medianos, ambientes de computo distribuidos que incluyen cualesquiera de los anteriores sistemas o dispositivos, etc. Versiones compactas o subconjuntos de la estructura también pueden ser implementadas en clientes de recursos limitados, tales como teléfonos celulares, asistentes digitales personales, computadores de mano, u otros dispositivos de comunicación/computo.

La estructura 132 puede ser descrita en el contexto general de instrucciones ejecutables por computadora, tal como módulos de programa, que son ejecutados por uno o más computadores u otros dispositivos. Generalmente, los módulos de programa incluyen rutinas, programas, objetos, componentes, estructuras de datos, etc., que llevan a cabo tareas particulares o implementan tipos de datos abstractos particulares. La estructura 132 puede también ser practicada en ambientes de cómputo distribuidos en

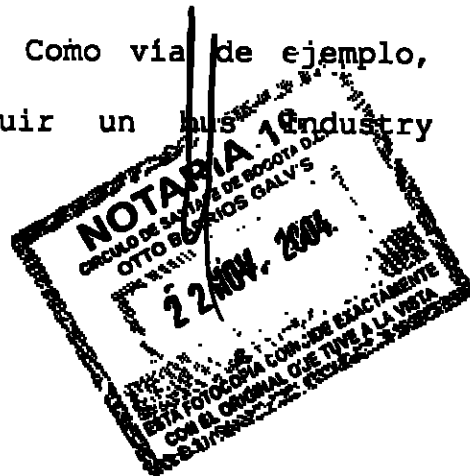


966

donde las tareas se llevan a cabo por dispositivos de procesamiento remotos que están enlazados por medio de redes de comunicaciones. En un ambiente de cómputo distribuido, los módulos de programa pueden estar localizados tanto en medios de almacenamiento de computadoras locales y remotas incluyendo dispositivos de almacenamiento en memoria.

El ambiente de cómputo 400 incluye un dispositivo de cómputo de propósito general en la forma de un computador 402. Los componentes del computador 402 pueden incluir, pero no limitarse a, uno o más procesadores o unidades de procesamiento 404, una memoria del sistema 406, y un bus del sistema 408 que acopla diversos componentes del sistema incluyendo el procesador 404 a la memoria del sistema 406.

El bus del sistema 408 representa uno o más de diversos tipos posibles de estructuras de bus, incluyendo un bus de memoria o controlador de memoria, un bus de periféricos, un puerto de gráficos acelerados, y un bus de procesador o bus local que usa cualquiera de una variedad de arquitecturas de bus. Como vía de ejemplo, tales arquitecturas pueden incluir un bus Industry

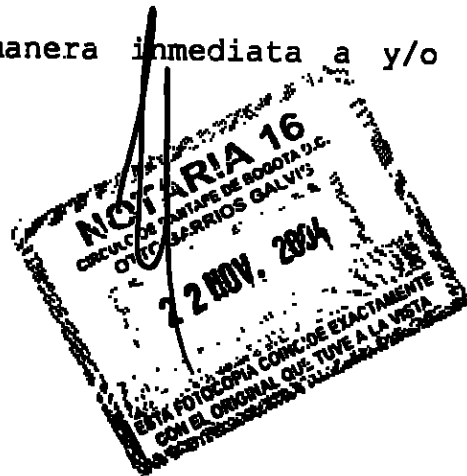


12
967

Standard Architecture (ISA), un bus de Arquitectura Micro Canal (MCA), un bus de ISA mejorado (EISA), un bus local de Video Electronics Standards Association (VESA), y un bus de Interconexión de Componentes Periféricos (PCI) también conocido como bus Mezanine.

En la computadora 402 típicamente incluye una variedad de medios legibles por computadora. Tales medios pueden ser cualquiera de los medios disponibles que son accesibles por la computadora 402 e incluye tanto medios volátiles como no volátiles medios removibles y medios no removibles.

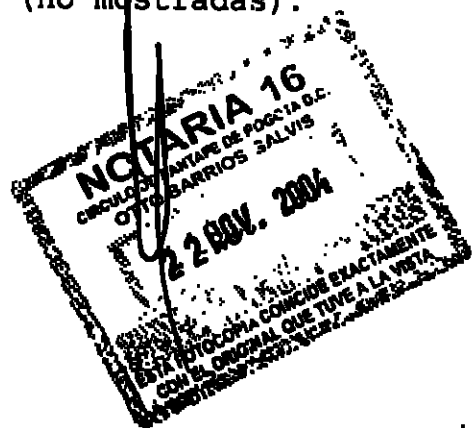
La memoria del sistema 406 incluye medios legibles por computadora en la forma de memoria volátil, tal como una memoria de acceso aleatorio (RAM) 410, y/o memoria no volátil tal como una memoria de solo lectura (ROM) 412. Un sistema básico de entrada/salida (BIOS) 414, contiene las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 402, tal como durante el inicio, se almacena en la ROM 412. La RAM 410 típicamente contiene datos y/o módulos de programa que son accesibles de manera inmediata a y/o



468

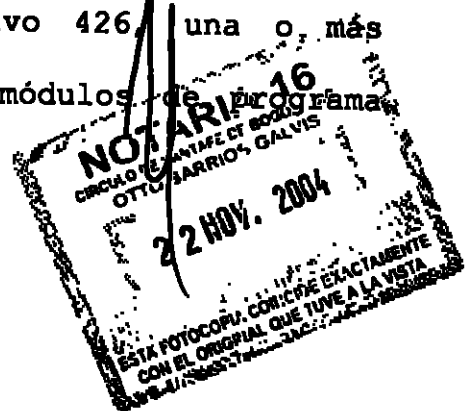
están operados actualmente por la unidad de procesamiento 404.

La computadora 402 puede también incluir otros medios de almacenamiento en computadora removibles/no removibles, volátiles/no volátiles. Como vía de ejemplo, la figura 4 ilustra un dispositivo de disco duro 416 para leer desde y escribir en un medio magnético no removible, no volátil (no mostrado) un dispositivo de disco magnético 418 para leer desde y escribir en un disco magnético removible, no volátil, 420 (por ejemplo, un disquete), y un dispositivo de disco óptico 422 para leer desde y/o escribir en un disco óptico removible, no volátil 424 tan como un CD-ROM, DVD-ROM, u otro medio óptico. El dispositivo de disco duro 416, el dispositivo de disco magnético 418, y el dispositivo de disco óptico 422 están cada uno conectados al bus del sistema 408 por una o más Interfaces de medios de datos 426. Alternativamente, el dispositivo de disco duro 416, el dispositivo de disco magnético 418, y el dispositivo de disco óptico 422 pueden estar conectados al bus del sistema 408 por una o más Interfaces (no mostradas).



Los dispositivos de disco y sus medios legibles por computadora asociados suministran almacenamiento no volátil de instrucciones legibles por computadora, estructuras de datos, módulos de programa, u otros datos para la computadora 402. Aunque el ejemplo ilustra un disco duro 416, un disco magnético removible 420, y un disco óptico removible 424, debe apreciarse que otros tipos de medios legibles por computadora que puedan almacenar datos que sean accesible por una computadora, tal como casetes magnéticos, u otros dispositivos de almacenamiento magnético, tarjetas de memoria flash, CD-ROM, discos versátiles digitales (DVD) u otros almacenamientos ópticos, memorias de acceso aleatorio (RAM), memorias de solo lectura (ROM), memorias de solo lecturas programables borrables eléctricamente (EEPROM), y similares, también pueden ser utilizados para implementar el sistema de cómputo ilustrativo y el ambiente.

Cualquier número de módulos de programa pueden ser almacenados sobre el disco duro 416, disco magnético 420, disco óptico 424, y ROM 412, y/o RAM 410, incluyendo por vía de ejemplo, un sistema operativo 426, una o más programas de aplicación 428, otros módulos de programa

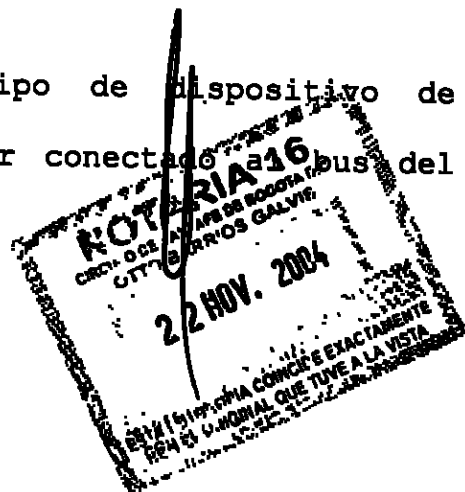


910

430, y datos de programa 432. Cada uno del sistema operativo 426, uno o más programas de aplicación 428, otros módulos de programa 430 y datos de programa 432 (o alguna combinación de ellos) puede incluir elementos de la estructura de programación 132.

Un usuario puede ingresar comandos e información hacia el computador 402 por medio de dispositivos de entrada tales como un teclado 434 y un dispositivo de apuntamiento 4436 (por ejemplo, un "ratón"). Otros dispositivos de entrada 438 (no mostrados específicamente) pueden incluir un micrófono, un joystick, una almohadilla de juego, un plato de satélite, un puerto serial, un scanner, y/o similares. Estos y otros dispositivos de entrada están conectados a la unidad de procesamiento 404 por medio de Interfaces de entrada/salida 440 que están acoplados al bus del sistema 408, que pero pueden estar conectados por otras Interfaces y estructuras de bus, tales como un puerto paralelo, un puerto de juegos, o un bus serial universal (USB).

Un monitor 442 u otro tipo de dispositivo de visualización puede también estar conectado al bus del

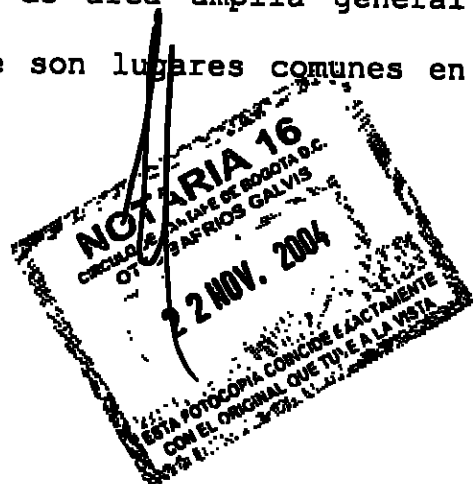


421

sistema 408 por medio de una interfaz, tal como un adaptador de video 444. En adición al monitor 442, otros dispositivos periféricos de salida pueden incluir componentes tales como parlantes (no mostrados) y una impresora 446 que puede estar conectada al computador 402 por medio de Interfaces de entrada/salida 440.

El computador 402 puede operar en un ambiente de red usando conexiones lógicas a uno o más computadores remotos, tal como el dispositivo de cómputo remoto 448. Por vía de ejemplo, el dispositivo de cómputo remoto 448 puede ser un computador personal, un computador portátil, un servidor, un enrutador, un computador de red, un dispositivo parejo u otro nodo de red común, etc. El dispositivo de cómputo remoto 448 es ilustrado como un computador portátil que puede incluir muchos o todos de los elementos y rasgos descritos aquí en relación al computador 402.

Las conexiones lógicas entre el computador 402 y el computador remoto 448 están ilustradas como una red de área local (LAN) 450 y una red de área amplia general (WAN) 452. Tales ambientes de red son lugares comunes en



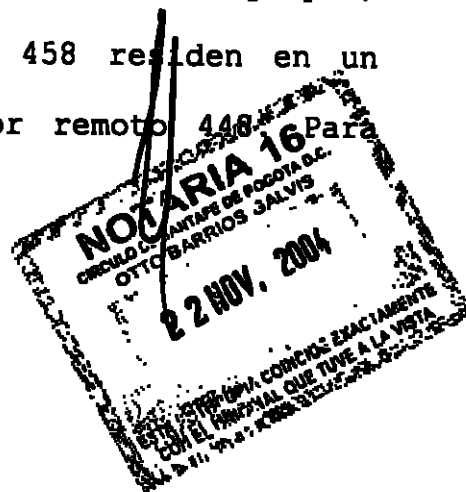
320

270
922

oficinas, redes de computador de empresas grandes, intranet y el Internet.

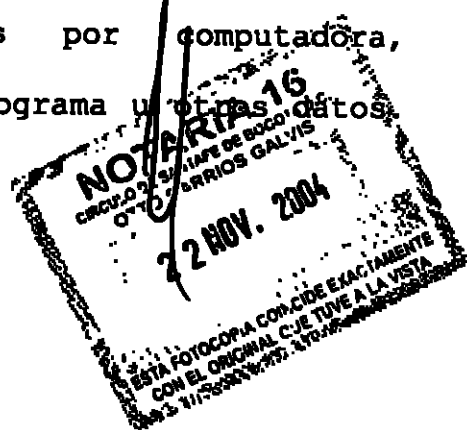
Cuando se ha implementado en un ambiente de red LAN, el computador 402 está conectado a una red local 450 por medio de una Interfaz de red o adaptador 454. Cuando se ha implementado en un ambiente de red WAN, el computador 402 típicamente incluye un módem 456 u otros medios para establecer comunicaciones sobre la red amplia 452. El módem 456, puede ser interno o externo al computador 402, puede estar conectado al bus del sistema 408 vía las Interfaces de entrada/salida 440 u otros mecanismos apropiados. Debe apreciarse que las conexiones de red ilustradas son ilustrativas y que otros medios de establecer comunicaciones o enlaces de comunicaciones entre los computadores 402 y 448 pueden ser empleados.

En un ambiente en red, tal como el ilustrado con el ambiente de computo 400, los módulos de programa ilustrados con relación al computador 402 o porciones de ellos pueden estar almacenados en un dispositivo d almacenamiento de memoria remoto. Como vía de ejemplo, los programas de aplicación remotos 458 residen en un dispositivo de memoria del computador remoto 448.



propósitos de ilustración, los programas de aplicación y otros componentes de programa ejecutables tales como el sistema operativo están lustrados aquí como bloques discretos, aunque debe reconocerse que tales programas y componentes residen a diversos tiempos en diferentes componentes de almacenamiento del dispositivo de computo 402, y son ejecutados por los procesadores de datos del computador.

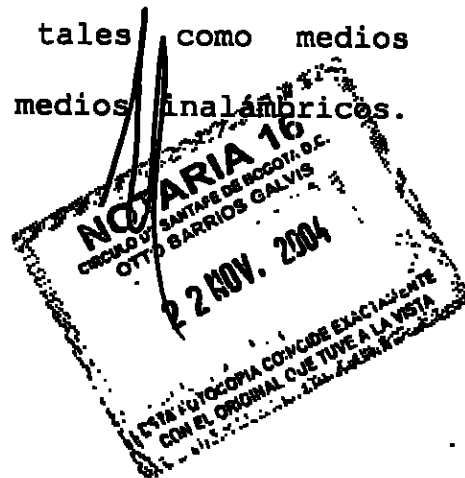
Una implementación de la estructura 132, y particularmente, el API 142 o las llamadas hechas al API 142, pueden ser almacenadas en o transmitidas por alguna forma de medio legible por computador. El medio legible por computador puede ser cualquiera de los medios disponibles que puedan ser accedados por una computadora. Como vía de ejemplo, y no como limitación, el medio legible por computador puede comprender "medio de almacenamiento por computador" y "medio de comunicaciones" el "medio de almacenamiento de computador" incluye medios volátiles y no volátiles, removibles y no removibles implementados por cualquiera método o tecnología para el almacenamiento de información tal como instrucciones legibles por computadora, estructuras de datos, módulos de programa u otros datos.



974

El medio de almacenamiento por computador incluye, pero no se limita a, RAM, EEPROM, memoria flash y otras tecnologías de memoria, CD-ROM, discos versátiles digitales (DVD), u otros almacenamientos ópticos, Casetes magnéticos, cinta magnética, almacenamiento de disco magnético, u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda ser utilizado para almacenar la información deseada y que pueda ser accesado por un computador.

El "medio de comunicación" típicamente involucra instrucciones legibles por computadora, estructuras de datos, módulos de programa, u otros datos en una señal de datos modulada, tal como una onda portadora u otro mecanismo de transporte. El medio de comunicación también incluye cualquier medio de suministro de información. El término "señal de datos modulada" significa una señal que tiene una o más de sus características establecidas o cambiadas de tal manera que se codifica la información en la señal. Como vía de ejemplo, y no como limitación, el medio de comunicación incluye medios alambrados tales como una red alambrada o una conexión directamente alambrada, y medios inalámbricos tales como medios acústicos, RF, infrarrojo y otros medios inalámbricos.

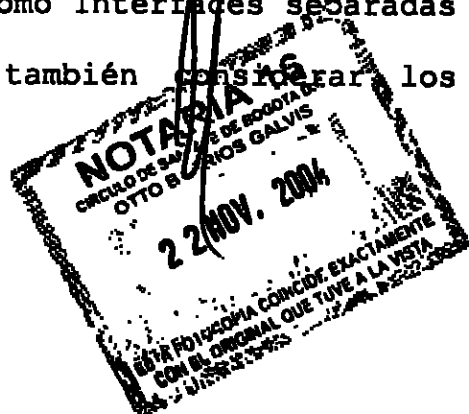


925

Combinaciones de cualquiera de los anteriores también están incluidas dentro del alcance del medio legible por computador.

Alternativamente, porciones de la estructura pueden ser implementadas en hardware o en una combinación de hardware, software, y/o firmware. Por ejemplo, una o más aplicaciones de circuitos integrados específicos (ASIC) o dispositivos lógicos programables (PLD) podrían ser diseñados o programados para implementar una o más porciones de la estructura.

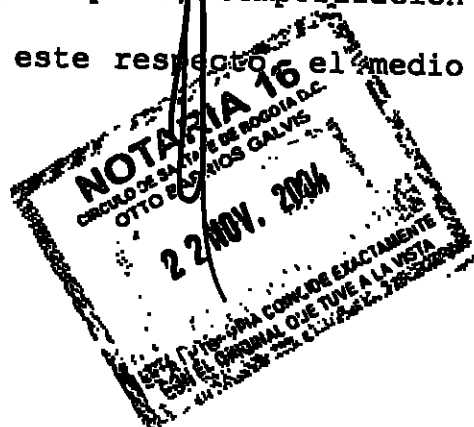
Notoriamente, una Interfaz de programación puede ser visualizada genéricamente, como se muestra en la figura 5 o figura 6. La figura 5 ilustra una Interfaz Interfaz 1 como un conducto por medio del cual primeros y segundos segmentos de código se comunican. La figura 6 ilustra una Interfaz según comprende la Interfaz e objetos I1 e I2 (que puede o no puede ser parte de primeros y segundos segmentos de código), que permiten primeros y segundos segmentos de código de un sistema se comuniquen vía el medio M. En la vista de la figura 6, uno puede considerar los objetos de Interfaz I1 y I2 como Interfaces separadas del mismo sistema y uno puede también considerar los



426

objetos I1 y I2 más el medio M que comprende la interfase. Aunque las figuras 5 y 6 muestran flujos e Interfaces bi-direccionales en cada lado del flujo, ciertas implementaciones pueden solamente tener información de flujo en una sola dirección (o ningún flujo de información como se describe más adelante) o puede solamente tener un objeto Interfaz sobre un lado. Como vía de ejemplo, y no como limitación, los términos tales como programación de aplicación o Interfaz de programa (API), punto de entrada, método, función, subrutina, llamada de procedimiento remoto, e Interfaz de modelo de objeto componente (COM), están involucradas o envueltas dentro de la definición de Interfaz de programación.

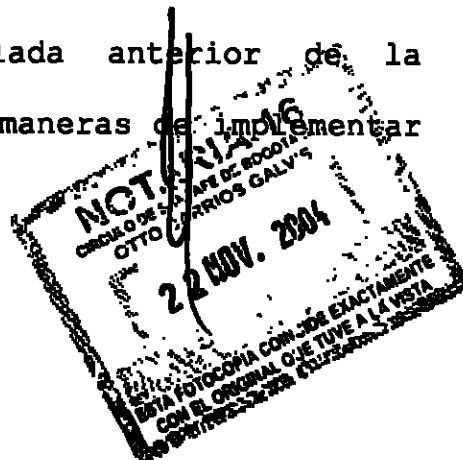
Aspectos de tal Interfaz de programación pueden incluir el método por medio del cual el primer segmento de código transmite información (en donde "información" es usada en su sentido más amplio e incluye datos, comandos, solicitudes, etc.) al segundo segmento de código; el método por medio del cual el segundo segmento de código recibe la información, y la estructura, secuencia, sintaxis, organización, esquema, temporización y contenido de la información. A este respecto el medio



927

de transporte subyacente mismo puede no ser importante a la operación de la interfase, si el medio es alambrado o inalámbrico, o una combinación de ambos, siempre que la información sea transportada de una manera definida por la interfase. En ciertas situaciones, la información puede no pasar en una o ambas direcciones en el sentido convencional, tal como la información se transfiere puede ser por medio de otros mecanismos (por ejemplo información colocada en una memoria, archivo, etc., separado del flujo de información entre los segmentos de código) o no existe, como cuando un segmento de código simplemente accesa la funcionalidad llevada a cabo por un segundo segmento de código. Cualquiera o todos de los aspectos pueden ser importantes en una situación dada, por ejemplo, dependiendo de si los segmentos de código son parte de un sistema en una configuración acoplada más libremente o en una configuración acoplada más estrictamente, de modo que esta lista debería considerarse ilustrativa y no limitativa.

Esta noción de una Interfaz de programación es conocida por aquellos basados en la técnica y es clara a partir de la descripción detallada anterior de la invención. Hay, sin embargo, otras maneras de implementar

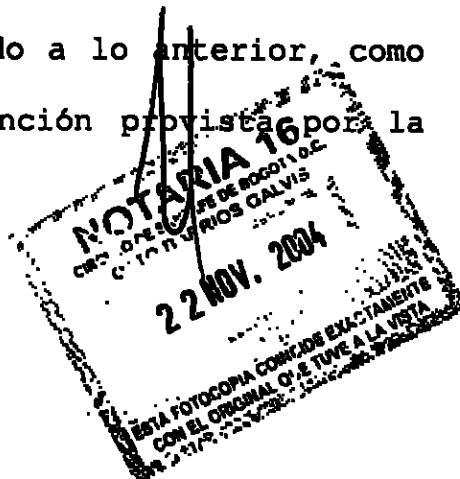


928

una Interfaz de programación, y, a menos que se excluya expresamente, estas también tienen el propósito de ser involucradas por las reivindicaciones establecidas al final de esta especificación tales otras maneras pueden aparecer como más sofisticadas o complejas que la vista simplística de las figuras 5 y 6, pero sin embargo llevan a cabo una función similar para lograr el mismo resultado general. Ahora brevemente se describe algunas implementaciones alternativas ilustrativas de una Interfaz de programación.

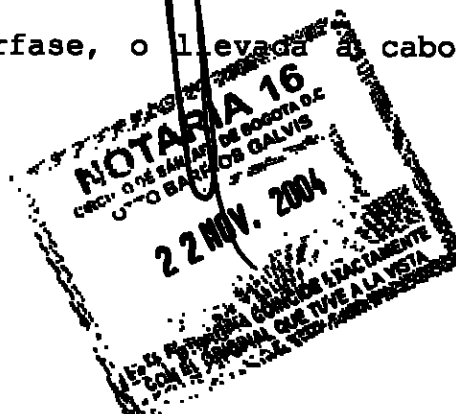
A. FACTORIZACIÓN

Una comunicación entre un segmento de código y otro puede llevarse a cabo indirectamente por medio de romper la comunicación en múltiples comunicaciones discretas. Esto es ilustrado esquemáticamente en las figuras 7 y 8. Como se muestra, algunas Interfaces pueden ser descritas en términos de conjuntos divisibles de funcionalidad. Así, la funcionalidad de Interfaz de las figuras 5 y 6 puede ser factorizada para lograr el mismo resultado, justo como no puede suministrar matemáticamente 24, o 2 veces 2 veces 3 veces 2. De acuerdo a lo anterior, como se ilustró en la figura 7, la función provista por la



429

Interfaz 6 Interfaz 1 puede estar subdividida para convertir las comunicaciones de la interfaz en Interfaces múltiples Interfaz 1A, Interfaz 1B, Interfaz 1C, etc. mientras se logra el mismo resultado. Como se ilustró en la figura 8, la función provista por Interfaz I1 puede estar subdividida en múltiples Interfaces I1a, I1b, I1c, etc. Mientras se logra el mismo resultado. Similarmente, la Interfaz I2 del segundo segmento de código que recibe la información del primer segmento de código puede estar factorizada en múltiples Interfaces y I2a, I2b, I2c, etc. Cuando se factoriza, el número de Interfaces incluidas con el primer segmento de código no requiere que case el número de Interfaces incluidas con el segundo segmento de código. En cualquiera de los casos de las figuras 7 y 8, el espíritu funcional de las Interfaces Interfaz 1 Y I1 permanece el mismo que en las figuras 5 y 6, respectivamente. La factorización de Interfaces puede también seguir las propiedades asociativa, conmutativa y otras de las matemáticas tal que la factorización pueda ser difícil de reconocer. Por ejemplo., la ordenación de operaciones puede no ser importante, y consecuentemente, una función llevada a cabo por una Interfaz puede llevar a cabo muy bien un avance en el alcance de la interfase, por otra pieza de código o interfase, o llevada a cabo

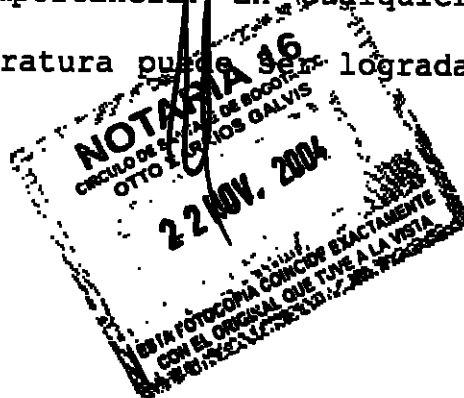


980

por un componente separado del sistema. Más aún, una persona versada en la materia en las técnicas de programación puede apreciar que hay una variedad de maneras de hacer diferentes llamadas de función que logran el mismo resultado.

B. REDEFINICIÓN

En algunos casos, puede ser posible ignorar, agregar o redefinir ciertos aspectos (por ejemplo parámetros) de una Interfaz de programación mientras que todavía se lleva a cabo el resultado propuesto. Esto se ilustra en las figuras 9 y 10. Por ejemplo, asuma que la Interfaz Interfaz 1 de la figura 5 incluye una función de llamada *Square(input, precision, output)*, una llamada que incluye tres parámetros, *input*, *precision* *output*, y que es emitida desde el primer segmento de código al segundo segmento de código, si el parámetro medio *precision* no es de importancia en un escenario dado, como se muestra en la figura 9, este justamente puede ser perfectamente bien ignorado o aun reemplazado con un parámetro *meaningless* (en esta situación). Uno puede también agregar un parámetro *additional* o de no importancia. En cualquier evento, la funcionalidad de cuadratura puede ser lograda

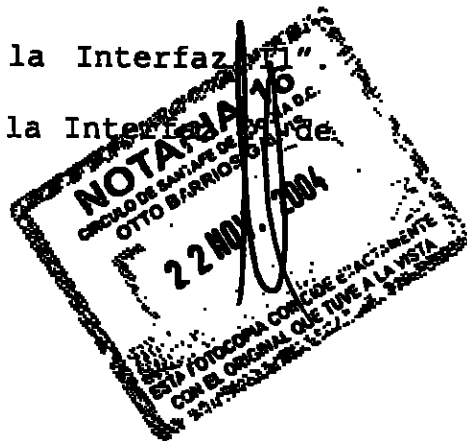


910

982

B. CODIFICACIÓN EN LÍNEA

Puede ser también factible fundir algunas o todas las funcionalidades de 2 módulos de código separados tales que la "interfase" entre ellos cambia de forma. Por ejemplo, la funcionalidad de las figuras 5 y 6 puede ser convertida a la funcionalidad de las figuras 11 y 12, respectivamente. En la figura 11, los primeros y segundos segmentos de código anteriores de la figura 5 son fundidos en un módulo que los contiene a ambos, En este caso, los segmentos de código todavía pueden comunicarse uno con el otro pero la Interfaz puede estar adaptada a una forma que es más adecuada a un módulo único. Así, por ejemplo, las aseveraciones Call y Return formales pueden ya no requerirse, pero procesamiento o respuesta similar siguiendo a la Interfaz Interfaz 1 puede todavía estar en efecto. Similarmente, mostrado en la figura 12, parte (o todo) de la Interfaz I2 de la figura 6 puede escribirse en línea en la Interfaz I1 para formar la Interfaz I1". Como se ha ilustrado, la Interfaz I2 está dividida en I2a e I2b, y la porción de Interfaz I2a ha sido codificada en línea con la Interfaz I1 para formar la Interfaz I1". Para un ejemplo concreto, considere que la Interfaz

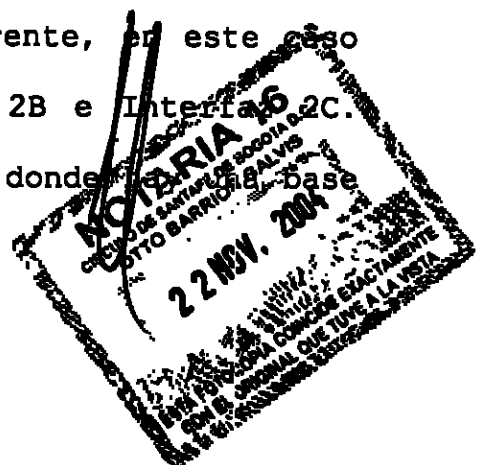


983

la figura 6 lleva a cabo una función de llamar cuadrado (*input*, *output*), que es recibida por la Interfaz I2, que después de procesar el valor pasado con *input* (para hacer el cuadrado) por el segundo segmento de código, lo pasa de nuevo el resultado cuadrado con *output*. En tal caso, el procesamiento llevado a cabo por el segundo segmento de código (hacer el cuadrado de *input*) puede ser llevado a cabo por el primer segmento de código sin que se llame a la interfase.

D. DIVORCIO

Una comunicación de un segmento de código a otro puede llevarse a cabo indirectamente rompiendo la comunicación en múltiples comunicaciones discretas. Esto se ha ilustrado esquemáticamente en las figuras 13 y 14. Como se muestra en la figura 13, una más piezas del programa medio (Divorce Interface(s), ya que ellos divorcian la funcionalidad y/o las funciones de Interfaz de la Interfaz original) se suministran para convertir las comunicaciones en una primera interfase, Interfase1, para conformarlas a una Interfaz diferente, en este caso las Interfaces Interfaz 2A, Interfaz 2B e Interfaz 2C. Esto puede ser hecho, por ejemplo, en donde



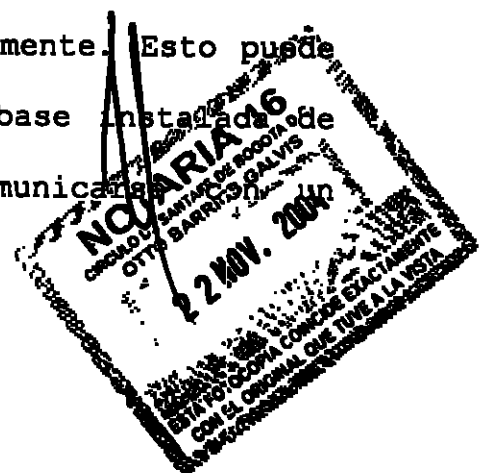
984

instalada de aplicaciones diseñadas para comunicarse con, digamos, un sistema operativo de acuerdo con un protocolo interfaz1, pero luego el sistema operativo es cambiado para usar una Interfaz diferente, en este caso las Interfaces Interfaz 2A, Interfaz 2B e Interfaz 2C. El punto es que la Interfaz original usada por el segundo segmento de código es cambiada de modo que ya no es más compatible con la Interfaz usada por el primer segmento de código, y por lo tanto un intermediario es usado para hacer que las Interfaces viejas y nuevas sean compatibles. Similarmente, como se muestra en la figura 14, un tercer segmento de código puede ser introducido con el divorcio de la Interfaz de DI1 para recibir las comunicaciones de la Interfaz I1 y con la Interfaz divorciada DI2 para transmitir la funcionalidad de Interfaz a, por ejemplo, las Interfaces I2a e I2b, rediseñadas para trabajar con DI2, pero que suministran el mismo resultado funcional. Similarmente, DI1 y DI2 pueden trabajar juntas para traducir la funcionalidad de las Interfaces y I1 e I2 de la figura 6 a un nuevo sistema operativo, aunque se suministra el mismo resultado funcional o uno similar.



E. REESCRITURA

Todavía otra posible variante es de reescribir dinámicamente el código para reemplazar la funcionalidad de la Interfaz con algo distinto pero que logra el mismo resultado general. Por ejemplo, puede haber un sistema en el cual un segmento de código presentado en un lenguaje intermedio (por ejemplo, Microsoft IL, Java ByteCode, etc.) esta provisto en un compilador de justo en tiempo (JIT) o interpretador en un ambiente de ejecución (tal como aquel provisto por la estructura .Net, el ambiente de corrida Java u otros ambiente tipos de corrida similares). El compilador JIT puede estar escrito de modo que convierte dinámicamente las comunicaciones del primer segmento de código al segundo segmento de código, es decir, para conformarlos a una Interfaz diferente como puede ser requerida por el segundo segmento de código (sea el original o un diferente segundo segmento de código). Esto está ilustrado en las figuras 15 y 16. Como puede verse en la figura 15, esta enseñanza es similar al escenario de divorcio descrito anteriormente. Esto puede ser hecho, por ejemplo, en donde una base instalada de aplicaciones están diseñadas para comunicarse



486

sistema operativo de acuerdo con un protocolo de Interfaz 1, pero luego el sistema operativo es cambiado para usar una Interfaz diferente. El compilador JIT podría ser usado para conformar las comunicaciones al aire desde las aplicaciones instaladas en base a la nueva Interfaz del sistema operativo. Como se ilustró en la figura 16, esta enseñanza de reescribir dinámicamente la Interfaz puede ser aplicada a factorial dinámicamente, o de otra manera alterar las Interfaces también.

También se nota que los escenarios descritos anteriormente para lograr el mismo resultado similar como una Interfaz vía modalidades alternativas puede también ser combinada de varias maneras, serialmente y/o en paralelo, o con otros códigos de intervención. Así, las modalidades alternativas presentadas anteriormente no son mutuamente exclusivas y pueden ser mezcladas, casadas o combinadas para producir el mismo o escenarios equivalentes a los escenarios genéricos presentados en las figuras 5 y 6. Esto también se ha notado que como con la mayoría de las construcciones de programación, hay otras maneras similares de lograr el mismo o funcionalidades similares de una Interfaz que no queda ser descrita aquí, pero sin embargo están representadas

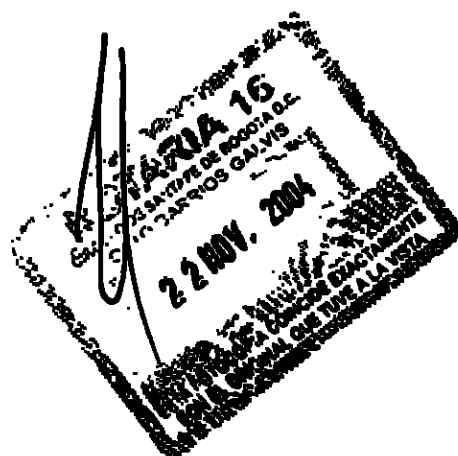


987

por el espíritu y alcance de la invención, es decir, se nota que hay por lo menos parcialmente la funcionalidad representada por los resultados ventajosos obtenidos de, una Interfaz que subyace en el valor de una interfase.

Conclusión

Aunque la invención ha sido descrita en un lenguaje específico a rasgos estructurales y /o actos metodológicos, debe entenderse que la invención definida en las reivindicaciones adjuntas no necesariamente se limita a rasgos específicos o actos descritos. En su lugar, los rasgos específicos y los actos específicos están descritos como formas ilustrativas de implementar la invención reivindicada.



REIVINDICACIONES

1. Una Interfaz de programación involucrada en uno o más medios legibles por computador, comprende:

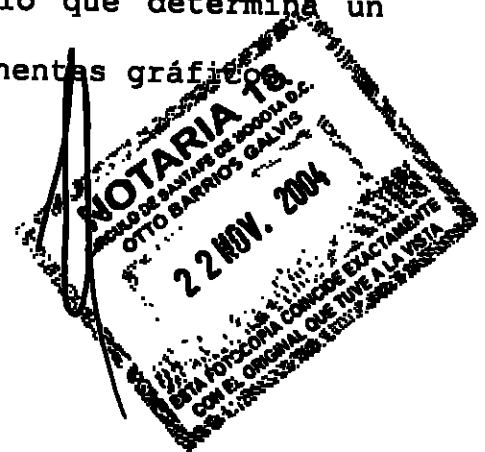
Un primer grupo de servicios relacionados para generar componentes gráficos;

Un segundo grupo de servicios relacionados para enlazar las propiedades de una clase a una fuente de datos; y

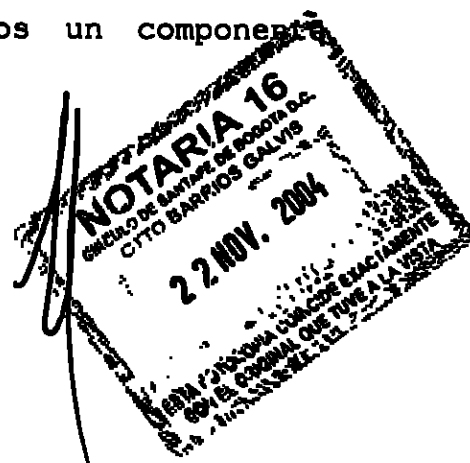
Un tercer grupo de servicios relacionados para formatear el contenido.

2. Una Interfaz de programación según la reivindicación 1, en donde el primer grupo de servicios incluye un servicio que determina una apariencia de los componentes gráficos.

3. Una Interfaz de programación según la reivindicación 1, en donde el primer grupo de servicio incluye un servicio que determina un comportamiento de los componentes gráficos.



4. Una Interfaz de programación según la reivindicación 1, en donde el primer grupo de servicios incluye un servicio que determina una disposición de los componentes gráficos.
5. Una Interfaz de programación según la reivindicación 1, en donde el primer grupo de servicios incluye una pluralidad de controles primitivos anidados que definen los componentes gráficos.
6. Una Interfaz de programación según la reivindicación 1, en donde los componentes gráficos están definidos por gráficos de vectores.
7. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionado con animación animar por lo menos un componente gráfico.



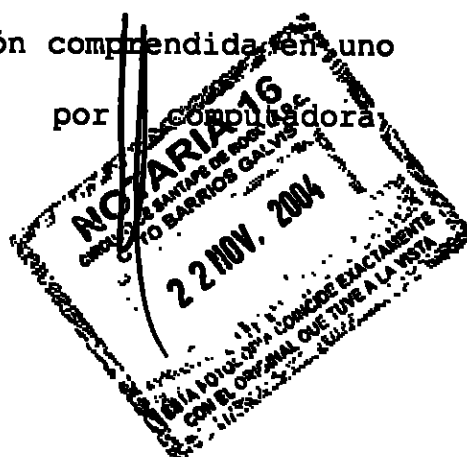
990

8. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionados con crear aplicaciones que tienen capacidades de navegación.
9. Una Interfaz de programación según la reivindicación 1 que comprende adicionalmente un cuarto grupo de servicios relacionados con soportar sistemas de procesamiento de tinta electrónica.
10. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionados con combinar una pluralidad de diferentes tipos de medios.
11. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionado con ejecutar aplicaciones sobre un cliente usando una Interfaz tipo buscador.



991

12. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionados con instalar aromáticamente y ejecutar una aplicación.
13. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionados con serializar contenido.
14. Una Interfaz de programación según la reivindicación 1, que comprende adicionalmente un cuarto grupo de servicios relacionados con automatizar la generación de una Interfaz de usuario.
15. Una arquitectura de software que comprende la Interfaz de programación como se mencionó en la reivindicación 1.
16. Una Interfaz de programación comprendida en uno o más medios legibles por computadora, comprende:



390

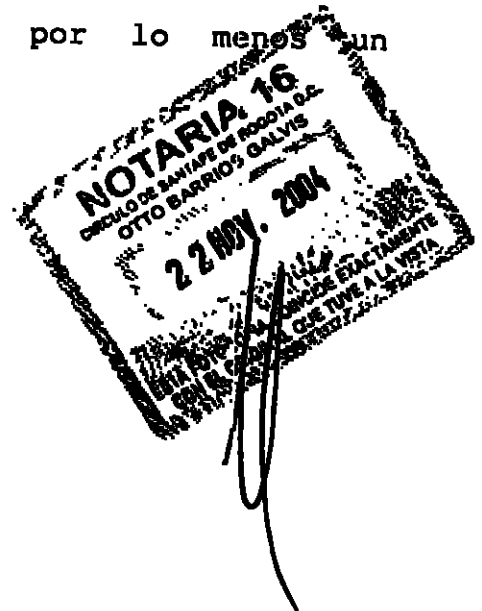
Un primer grupo de servicios relacionado con formatear el contenido antes de visualizar el contenido;

Un segundo grupo de servicios relacionados con ligar propiedades de una clase a una fuente de datos; y;

Un tercer grupo de servicios relacionados con generar efectos de imagen.

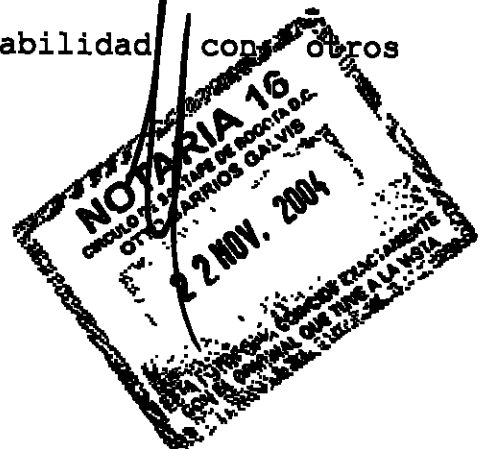
17. Una Interfaz de programación según la reivindicación 16, en donde el primer grupo de servicios incluye disponer una pluralidad de elementos de datos.

18. Una Interfaz de programación según la reivindicación 16, en donde el tercer grupo de servicios incluye animar por lo menos un elemento gráfico.



993

19. Una Interfaz de programación según la reivindicación 16, que comprende adicionalmente un cuarto grupo de servicios relacionados con crear aplicaciones que permitan a un usuario de la aplicación navegar entre una pluralidad de imágenes.
20. Una Interfaz de programación según la reivindicación 16, que comprende adicionalmente un cuarto grupo de servicio relacionados con editar previamente el contenido creado..
21. Una Interfaz de programación según la reivindicación 16, que comprende adicionalmente un cuarto grupo de servicio relacionado con administrar la entrada recibida desde un dispositivo de entrada.
22. Una Interfaz de programación según la reivindicación 16, que comprende adicionalmente un cuarto grupo de servicios relacionados con permitir la inter operabilidad con otros sistemas de cómputo.



312

994

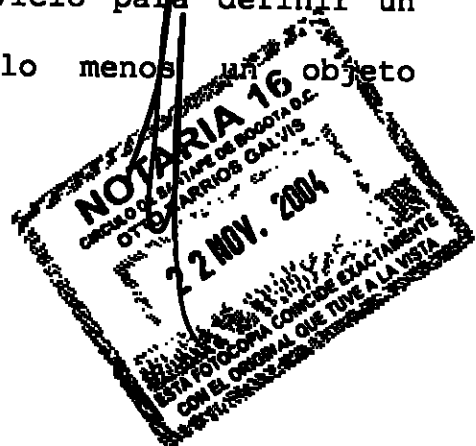
23. Un sistema de computador que incluye uno o más microprocesadores y uno o más programas de software, el uno o más programas de software utiliza una Interfaz de programa de aplicación para solicitar servicios de un sistema operativo, la Interfaz de programas de aplicación incluye comandos separados para solicitar los servicios que comprenden los siguientes grupos de servicios:

Un primer grupo de servicios relacionados con generar objetos gráficos;

Un segundo grupo de servicios relacionados con crear componentes de los objetos gráficos; y

Un tercer grupo de servicios relacionado con modificar una apariencia de los objetos gráficos.

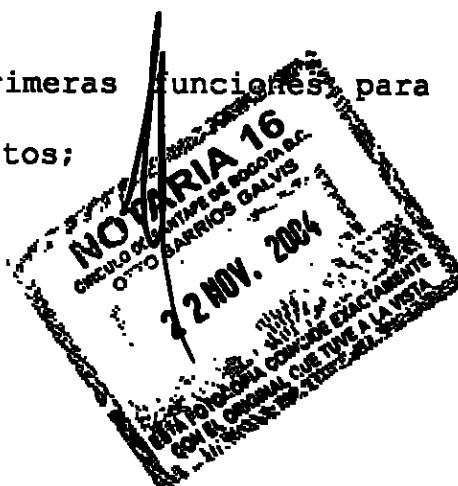
24. Un sistema de computador según la reivindicación 23, en donde el primer grupo de servicios incluye un servicio para definir un comportamiento de por lo menos un objeto gráfico.



323

995

25. Un sistema de computador según la reivindicación 23, en donde el primer grupo de servicios incluye un servicio para definir la disposición de objetos gráficos.
26. Un sistema de computador según la reivindicación 23, en donde modificar una apariencia de los objetos gráficos incluye animar los objetos gráficos.
27. Un sistema de computador según la reivindicación 23, en donde el segundo grupo de servicios incluye servicios para generar formas geométricas.
28. Un sistema de computador según la reivindicación 23, en donde la Interfaz de programa de aplicación incluye adicionalmente un cuarto grupo de servicios relacionados con formatear texto.
29. Un método comprende:
Llamar una o más de primeras funciones para facilitar el formatear datos;



299

996

Lllamar una o más de segundas funciones para facilitar la creación de objetos gráficos; y
Lllamar uno o más de terceras funciones para facilitar cambiar una apariencia de los objetos gráficos.

30. Un método según la reivindicación 29, que incluye adicionalmente llamar una o más de cuartas funciones para facilitar generar una Interfaz de usuario que usa una pluralidad de objetos gráficos.
31. Un método según la reivindicación 29, que incluye adicionalmente llamar una o más de cuartas funciones para facilitar la creación en tiempo de corrida de una Interfaz de usuario.
32. Un método según la reivindicación 29, que incluye adicionalmente:
Lllamar una o más de cuartas funciones para facilitar generar una Interfaz de usuario usando una pluralidad de objetos gráficos.



298

997

Llamar una o más de quintas funciones para facilitar la creación en tiempo de corrida de la Interfaz de usuario.

33. Un método según la reivindicación 29, en donde las primeras funciones facilitan:

Recibir la entrada de usuario; y

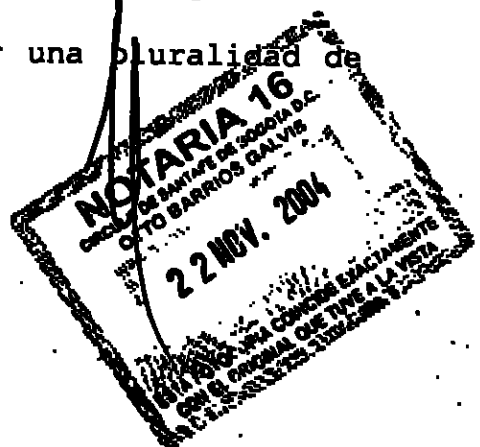
Disponer o arreglar los elementos de datos en una pantalla.

34. Un método según la reivindicación 29, en donde las segundas funciones facilitan generar formas geométricas.

35. Aun método según la reivindicación 29, en donde las segundas funciones facilitan generar por lo menos una forma geométrica y las terceras funciones facilitan modificar una apariencia de la forma geométrica.

36. Un sistema comprende:

medios para exponer un primer conjunto de funciones que permiten crear una pluralidad de formas geométricas;

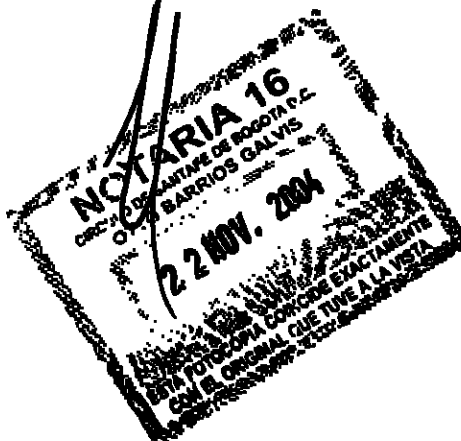


308

998

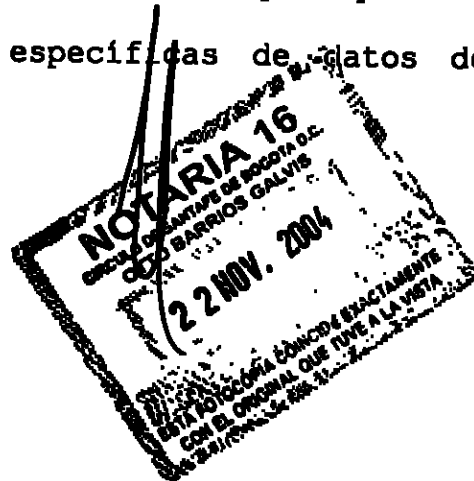
medios para exponer un segundo conjunto de funciones que permiten cambiar la manera en la cual las formas geométricas están dispuestas; y medios para exponer un tercer conjunto de funciones que permiten modificar apariencias de las formas geométricas.

37. Un sistema según la reivindicación 36, en donde el segundo conjunto de funciones permite adicionalmente disponer de las formas geométricas en una página para ser convertidas.
38. Un sistema según la reivindicación 36, en donde la pluralidad de formas geométricas incluye una línea.
39. Un sistema según la reivindicación 36, en donde el tercer conjunto de funciones permite adicionalmente asociar efectos de imagen con por lo menos una forma geométrica.



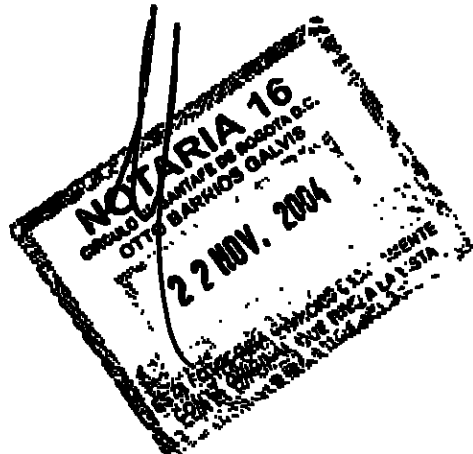
397

40. Un sistema según la reivindicación 36, en donde el tercer conjunto de funciones permite adicionalmente cambiar una apariencia de una forma geométrica particular en un periodo de tiempo.
41. Un sistema según la reivindicación 36, que comprende adicionalmente medios para exponer un cuarto conjunto de funciones que permiten la generación de una Interfaz de usuario que usa la pluralidad de formas geométricas.
42. Un sistema según la reivindicación 36, que comprende adicionalmente medios para exponer un cuarto conjunto de funciones que permiten asociar un objeto gráfico con una o más fuentes de datos.
43. Un sistema según la reivindicación 36, que comprende adicionalmente medios para exponer un cuarto conjunto de funciones que permite visualizar versiones específicas de datos de estos gráficos.



1000

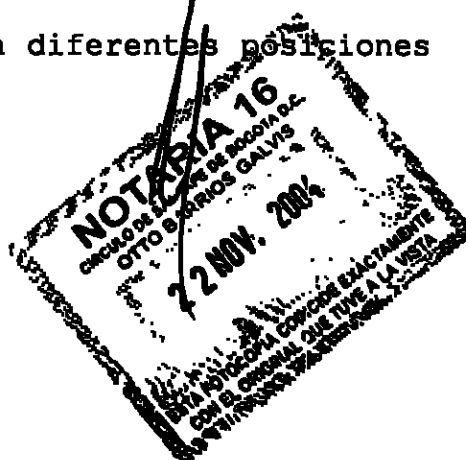
44. Un método comprende:
- Llamar una o más de primeras funciones para facilitar crear componentes de objetos gráficos;
- Llamar una o más de segundas funciones para facilitar generar objetos gráficos;
- Llamar una o más de terceras funciones para facilitar modificar una apariencia de objetos gráficos;
- Llamar una o más de cuartas funciones para facilitar la exposición de los objetos gráficos; y
- Llamar una más de quintas funciones para facilitar el asociar los objetos gráficos con fuentes de datos.
45. Un método según la reivindicación 44, que comprende adicionalmente llamar una o más de sextas funciones para facilitar navegar entre una pluralidad de pantallas de contenido.



399

1001

46. Un método según la reivindicación 44, en donde los componentes de los objetos gráficos incluyen una pluralidad de formas.
47. Un método según la reivindicación 44, en donde las segundas funciones facilitan adicionalmente generar una Interfaz de usuario gráfico.
48. Un método según la reivindicación 44, en donde las terceras funciones incluyen funciones que modifican la apariencia de un objeto gráfico particular.
49. Un método según la reivindicación 44, en donde las terceras funciones incluyen funciones que modifican la apariencia de uno o más componentes de un objeto gráfico.
50. Un método según la reivindicación 44, en donde las terceras funciones incluyen funciones que mueven objetos gráficos a diferentes posiciones en una pantalla.



1002

1000

1002

51. Un método según la reivindicación 44, en donde las terceras funciones modifican una apariencia de un objeto gráfico en respuesta a la entrada de usuario.

52. Un método según la reivindicación 44, en donde las cuartas funciones modifican una disposición de los objetos gráficos en respuesta a la entrada de usuario.

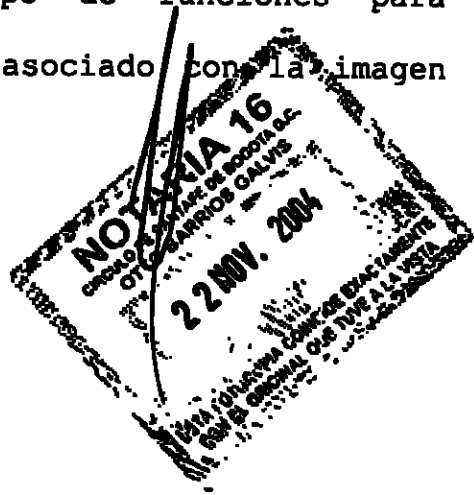
53. Un método para desarrollar un programa, el método comprende:

accesar un primer grupo de funciones para seleccionar componentes de una imagen gráfica;

accesar un segundo grupo de funciones para generar la imagen gráfica;

accesar un tercer grupo de funciones para modificar una apariencia de por lo menos un componente de la imagen gráfica; y

accesar un cuarto grupo de funciones para formatear el contenido asociado con la imagen gráfica.



401

54. Un método según la reivindicación 53, en donde la imagen gráfica es una Interfaz de usuario.
55. Un método según la reivindicación 53, en donde el contenido es información textual.
56. Un método según la reivindicación 53, en donde el contenido son datos gráficos.
57. Un método según la reivindicación 53, que comprende adicionalmente acceder un quinto grupo de funciones para asociar n componente de imagen gráfica con una fuente de datos.
58. Un método según la reivindicación 53, en donde el tercer grupo de funciones modifica adicionalmente un comportamiento asociado con el por lo menos un componente de la imagen gráfica.

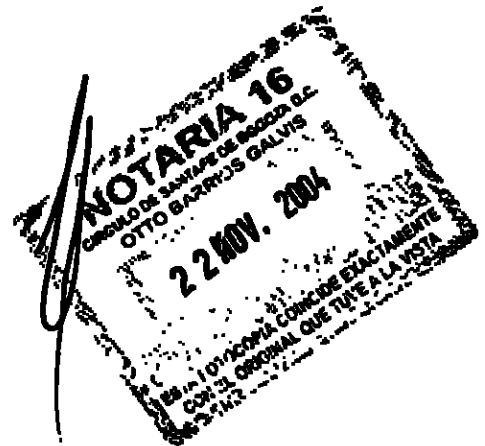


1002

1004

RESUMEN

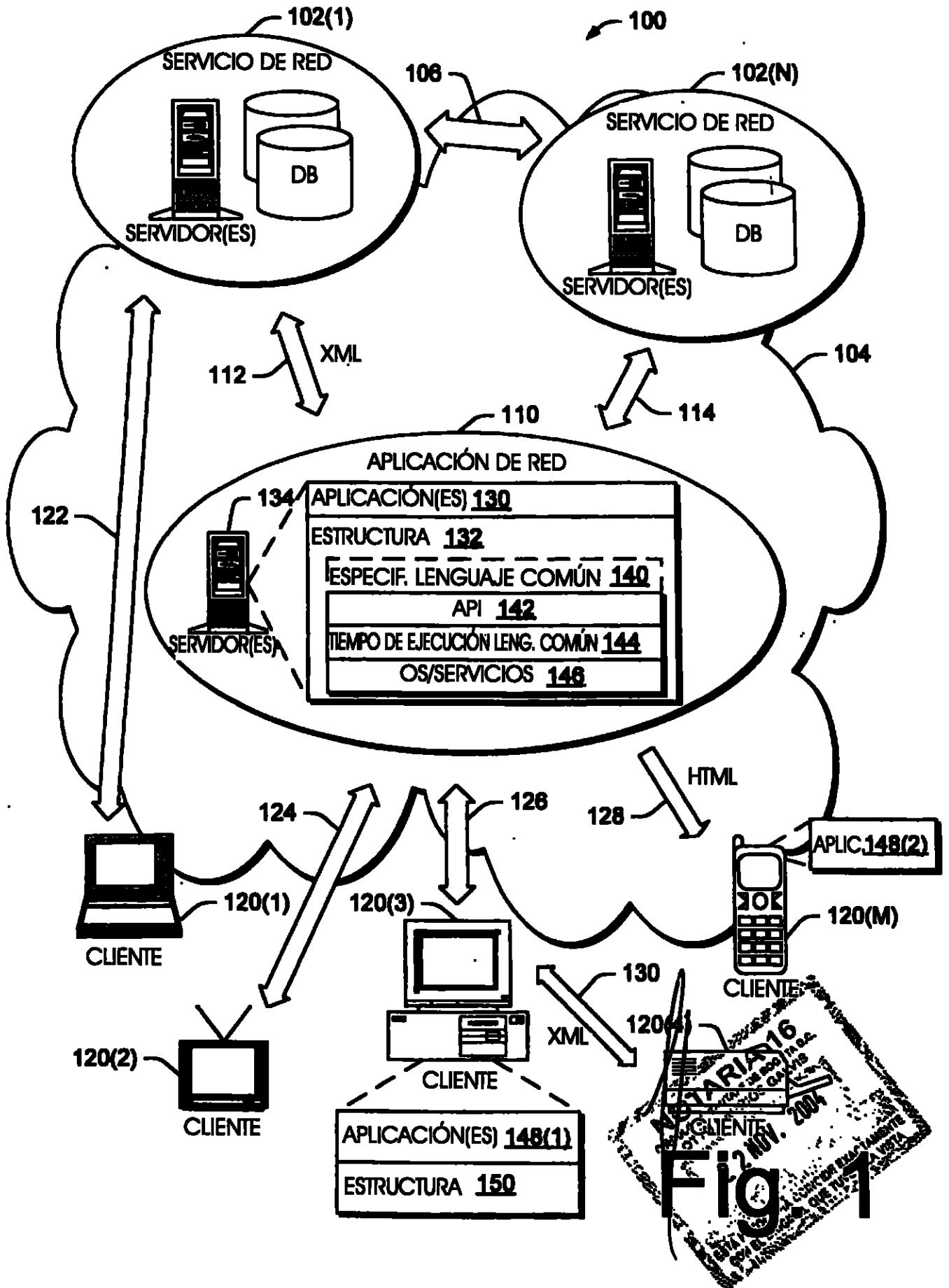
Una Interfaz de programación suministra funciones para generar aplicaciones, documentos, presentaciones de medios y otros contenidos. Estas funciones permiten a los desarrolladores obtener servicios de un sistema operativo, servicios de modelo de objeto, u otro sistema o servicio.



403

1005

1005



404

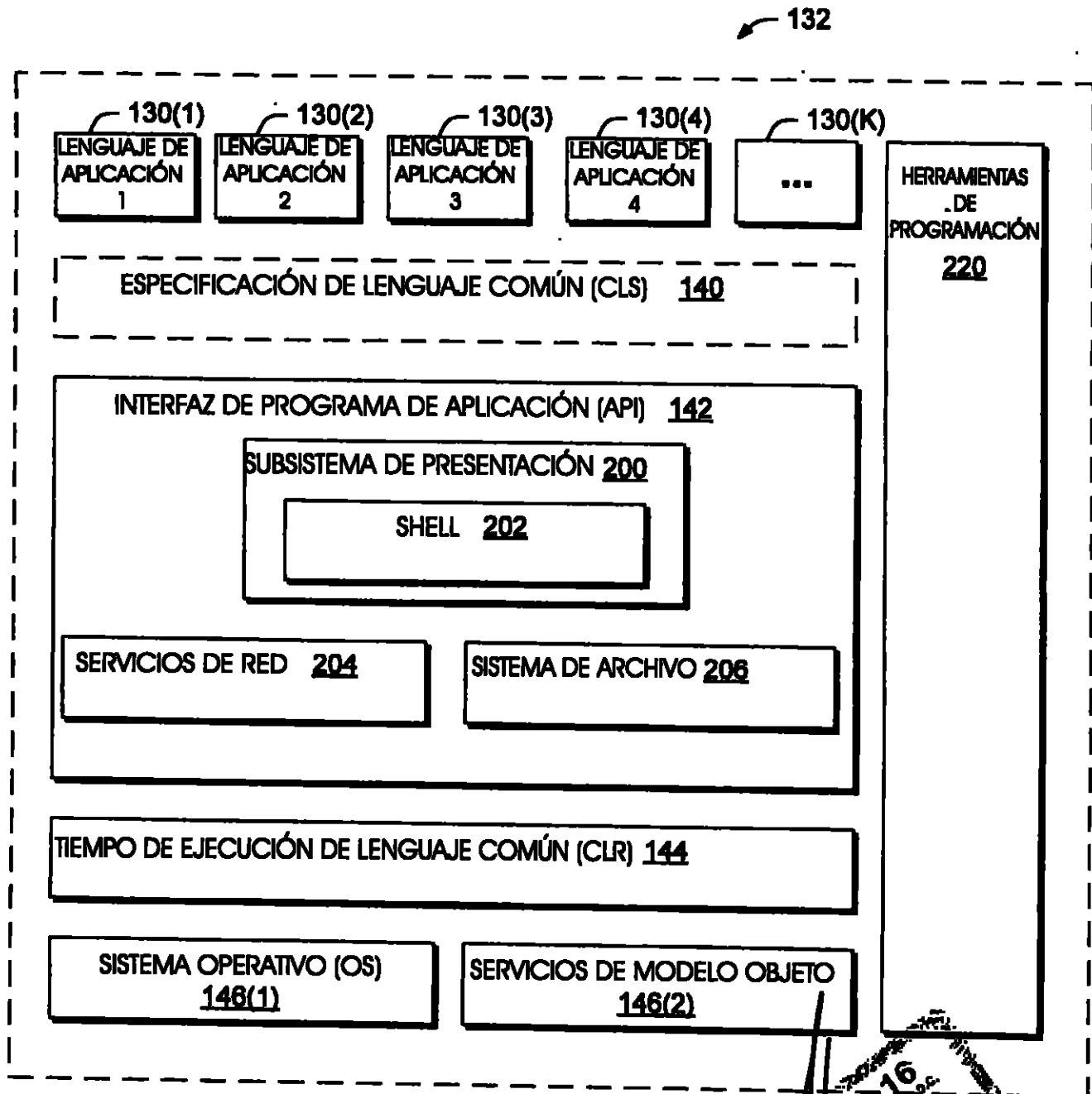
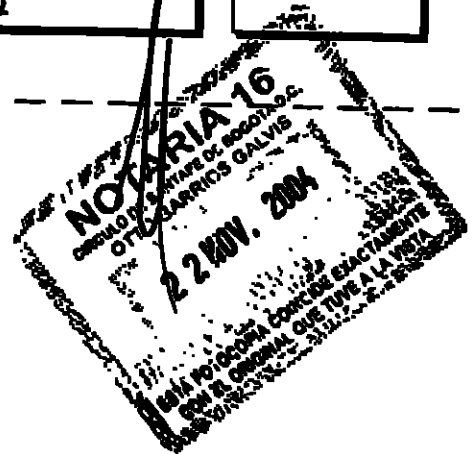


Fig. 2



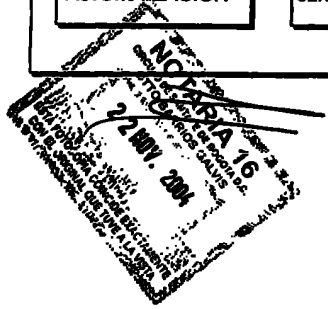
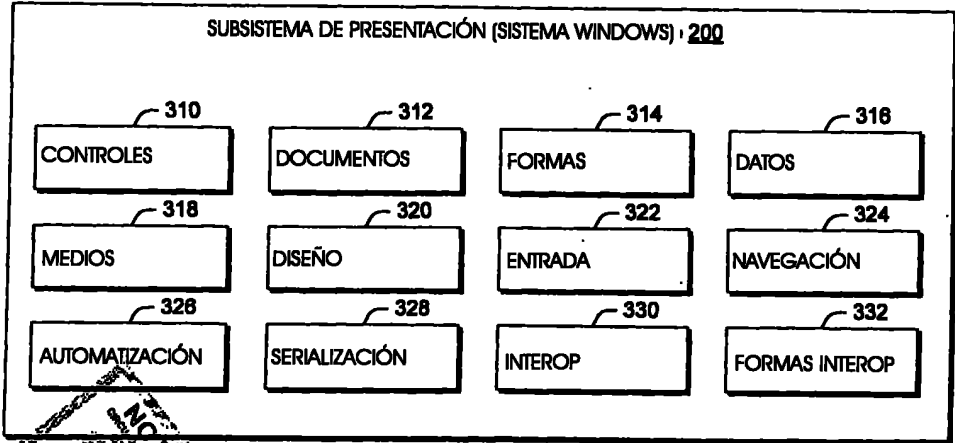


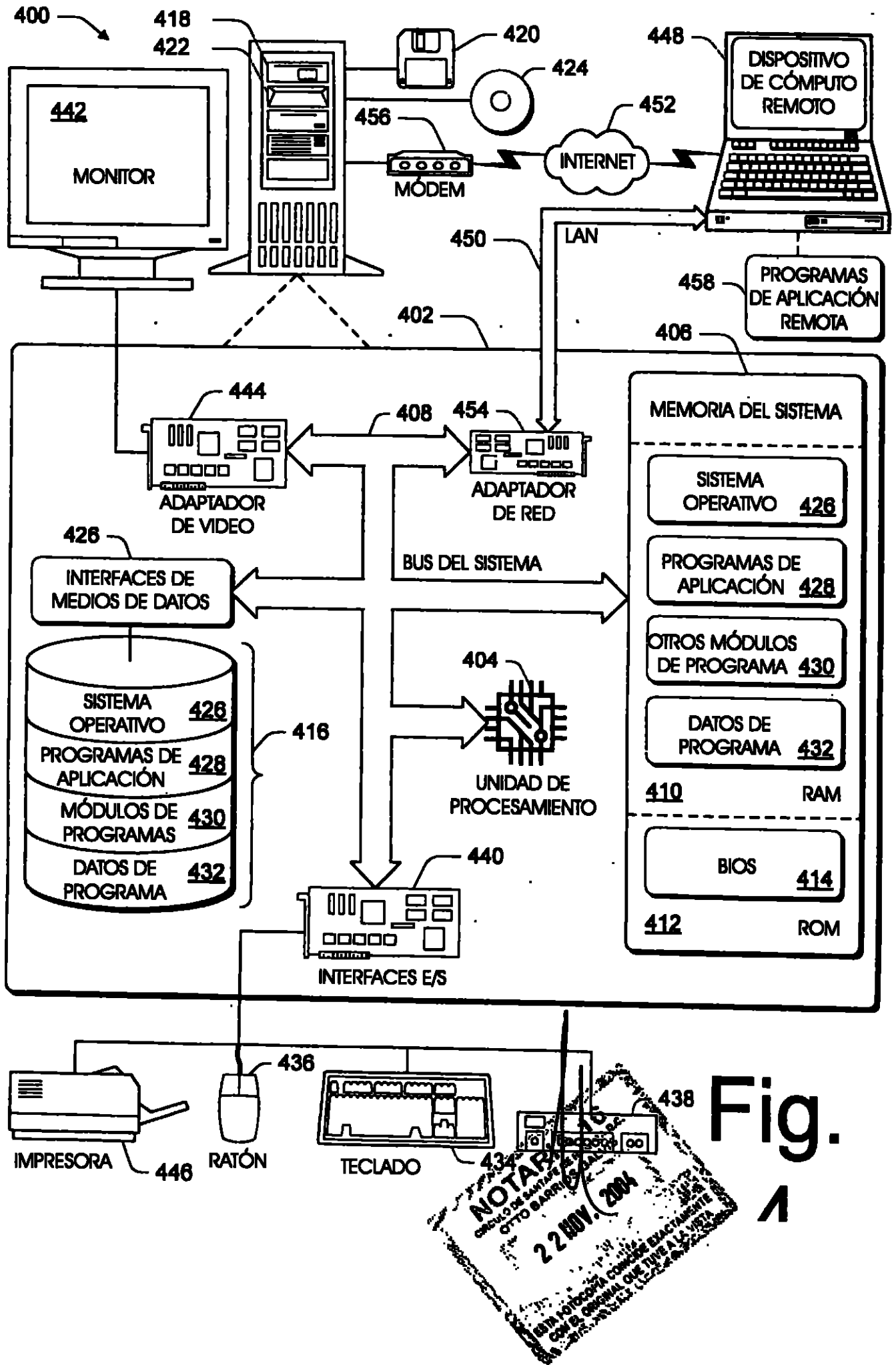
Fig.
3

907

1007

1066

1008



409

~~1007~~

1009

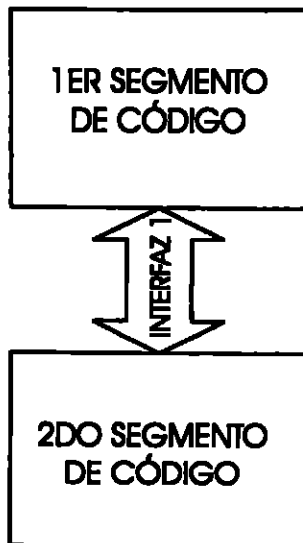


FIGURA 5

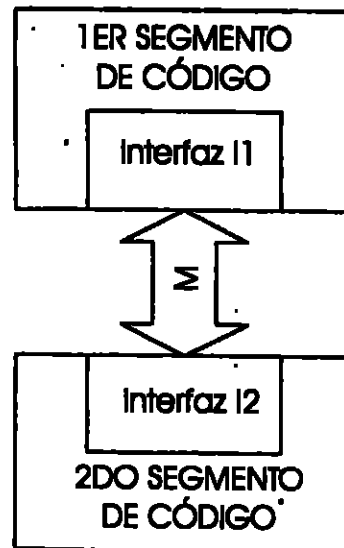


FIGURA 6

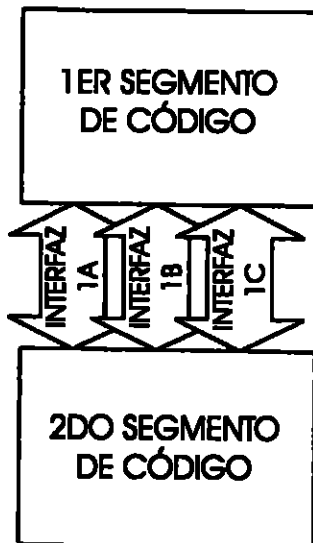


FIGURA 7

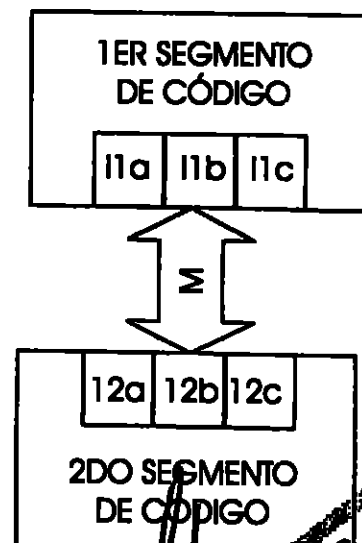


FIGURA 8

1079

1071

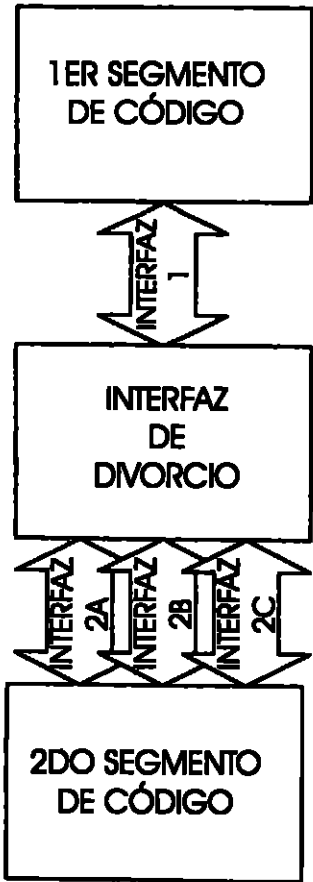


FIGURA 13

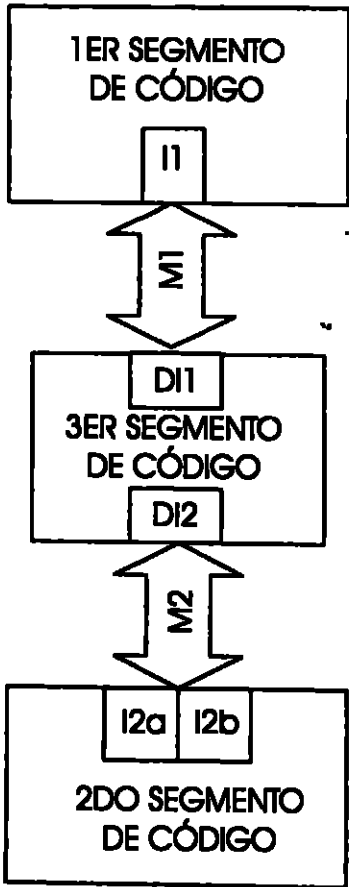
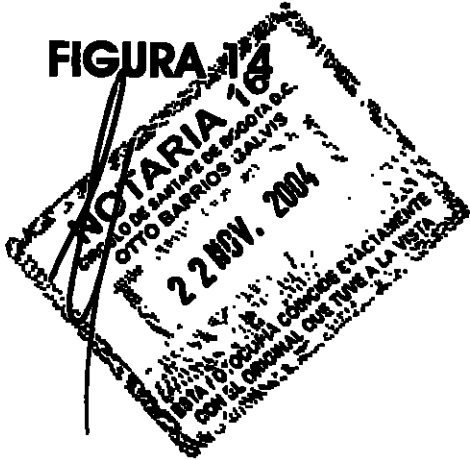


FIGURA 14



MSI-1779US

**FIGURA 15**

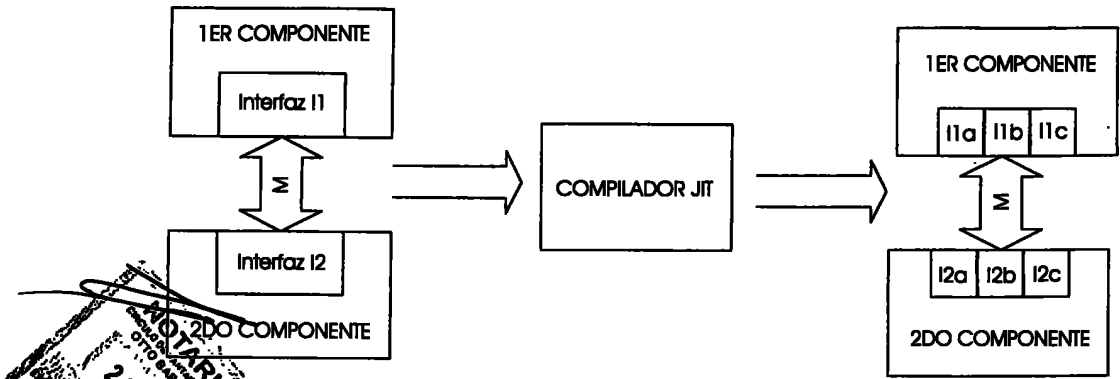


FIGURA 16

NOTARIA 16
22 NOV. 2004

24/12
000 E

1013

2° EXAMEN DE FORMA, PATENTE DE INVENCION

Radicación N° **04-90117**

Folio 1012

Concepto Técnico N° **705**

Clasificación Internacional de Patentes: **G01F 17/60**

Practicado el examen de acuerdo a lo establecido en el Art. 38 de la Decisión 486, se determinó que esta solicitud de **Patente**:

- SI ☒ NO ☐ Contiene un resumen de la invención (Art. 26-e)
- SI ☒ NO ☐ Contiene el título o nombre de la invención (Art. 27-d)
- SI ☒ NO ☐ Contiene la descripción de la invención (Art. 26-b)
- SI ☒ NO ☐ NO ES APLICABLE ☐ Contiene los dibujos necesarios para comprender la invención (Art. 26-d)
- SI ☒ NO ☐ Contiene una o más reivindicaciones (Art. 26-c)
- SI ☐ NO ☐ NO ES APLICABLE ☒ Han sido desarrollados los productos a partir de recursos genéticos o de sus productos derivados de los que cualquiera de los Países Miembros es país de origen (Art. 26-h)
- SI ☐ NO ☐ NO ES APLICABLE ☒ Los productos y o procedimientos han sido obtenidos o desarrollados a partir a partir de los conocimientos tradicionales de las comunidades indígenas, afro americanas, o locales de los Países Miembros, de acuerdo a lo establecido en la Decisión 391 y sus modificaciones y reglamentaciones vigentes (Art. 26-i)
- SI ☐ NO ☒ La invención se refiere a un producto relativo a un material biológico (Art. 26-j)
- SI ☒ NO ☐ NO ES APLICABLE ☐ La prioridad coincide con la materia reivindicada.
- SI ☒ NO ☐ **CUMPLE LOS REQUISITOS TÉCNICOS**

OBSERVACIONES Y OTROS: SI ☐ NO ☐ Ver hoja(s) anexa(s)

EXAMINADOR TÉCNICO: 202051

Bogotá D C, septiembre 5 de 2005.

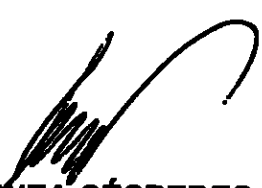
2020
Bogotá D C

Doctor(a):
DARIO CARDENAS NAVAS.
Apoderado (a)

REFERENCIA:	Radicación N°	04-90117
	Trámite	002
	Evento	001
	Actuación	416
	Oficio N°	860
	Folios	001

Teniendo en cuenta que la solicitud de privilegio de patente que se tramita bajo el expediente indicado en la referencia, reúne los requisitos de forma establecidos en los artículos 26 y 27 de la Decisión 486, de la Comisión de la Comunidad Andina, y en las demás disposiciones vigentes sobre la materia, se envía el extracto de la solicitud a la oficina de comunicaciones para efecto de su publicación en la Gaceta de Propiedad Industrial, conforme lo establece el artículo 40 de la Decisión 486, a partir del 10 de marzo de 2006.

Cúmplase
Dado en Bogotá D C,


ALIX CARMENZA CÉSPEDES DE VERGEL
Jefe División de Nuevas Creaciones.

202051